НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра Автоматики і управління у технічних системах

«На правах рукопису»
УДК _____

«Допущений до захисту»
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)
«____» _____ 2019 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 126 Інформаційні системи та технології
(код і назва спеціальності)

на тему: Система інтеграції гетерогенних джерел даних

Виконав студент II курсу, групи ІА-82мпв
(шифр групи)

Власов Владислав Володимирович _____
(прізвище, ім'я, по-батькові) (підпис)

Керівник зав. каф., д.т.н., проф. АУТС Ролік О.І. _____
(посада, наукова ступінь, звання, прізвище, ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, наукова ступінь, звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, наукова ступінь, вчене звання, прізвище, ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент_____
(підпис)

Київ – 2019 р.

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

ФІОТ

Кафедра АУТС

Ступінь вищої освіти «магістр»

Зі спеціальності 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ролік А.І.
(підпис) (ініціали, прізвище)
«__» _____.

ЗАВДАННЯ
на магістерську дисертацію студенту
Власову Владиславу Володимировичу .
(прізвище, ім'я, по батькові)

1. Тема роботи: Система інтеграції гетерогенних джерел даних
_____,

керівник роботи д.т.н. професор кафедри АУТС Ролік Олександр Іванович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ р. №

2. Строк подання студентом проекту_____

3. Вихідні дані до магістерської дисертації_____
_____
_____

4. Перелік завдань, які потрібно розробити: огляд, аналіз основних інтеграційних платформ, дослідження можливостей WSO2 EI, розробка структурної та функціональної схеми системи інтеграції, процес створення та розгортання інтеграційної системи

5. Перелік графічного матеріалу функціональна та структурна схема інтеграційної системи, діаграма процесу системи інтеграції, блок-схеми життєвого циклу розробки інтеграційної системи, гарантованої доставки повідомлень, вхідної,

вихідної послідовності та послідовності обробки помилок

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 05.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів проекту	Примітка
	Огляд, аналіз та опис предметної галузі і готових рішень	28.10.2019	
	Дослідження можливостей вибраної платформи	06.11.2019	
	Розробка структурної та функціональної схеми	14.11.2019	
	Процес створення на розгортання системи інтеграції	22.11.2019	
	Проведення аналізу стартап-проекту	25.11.2019	
	Оформлення ПЗ	03.12.2019	

Студент _____ Власов В.В.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Ролік О.І.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація освітньо-кваліфікаційного рівня “магістр” на тему: «Система інтеграції гетерогенних джерел даних». Робота включає 92 сторінки.

Магістерська дисертація містить результати розробки системи інтеграції гетерогенних джерел даних, яку можна використати як основу для реалізації аналогічних систем. Використано стандартні модулі платформи WSO2. Продемонстрована взаємодія сервісів всередині інтеграційної платформи.

Ключові слова: система інтеграції, інтеграція, гетерогенні джерела даних, трансформація даних

ABSTRACT

Master's dissertation of the educational qualification level "Master" on the topic: "Integration system of heterogeneous data sources". The work includes 92 pages.

The master's dissertation contains the results of the development integration system of heterogeneous data sources, which can be used as a basis for the implementation of similar solutions. Basic modules of the WSO2 integration platform were used. The interaction of services inside the platform was demonstrated.

Key words: integration system, integration, heterogeneous data sources, data transformation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС – інформаційні системи

ІТ – інформаційні технології

ПЗ – програмне забеспечення

API-M – API Manager

CLI – Command Line Interface

DAS – Data Analytics Server

DPG – DataPower gateway

EI – Enterpise Integrator

ESB – Entrerprise Service Bus

IoT – Internet of Things

IP – Internet Protocol

IS – Identity Server

JDK – Java development kit

JSON – JavaScript Object Notation

MI – Micro Integrator

MSF4J – Microservices Framework for Java

MQ – Message Queue

NoSQL – non SQL

SOA – Service-oriented architecture

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

REST – Representational State Transfer

VM – Virtual Machine

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

XML – Extensible Markup Languag

ВСТУП

На сьогоднішній день інформаційне середовище підприємства забезпечують і обслуговують інформаційні системи (ІС). При чому, підприємство не обмежується використанням єдиної корпоративної ІС (КІС). Так як ні одна КІС не здатна в повній мірі задовольнити всіх потреб підприємства в функціоналі: облік зарплати, управління персоналом, бухгалтерія, управління основним бізнес-процесом підприємства.

Але це не означає, що ІС, що використовуються на підприємстві оперують різними даними. Навпаки, ІС оперують повторюваним сегментом даних, тобто певні сегменти даних повторюються в кожній або в деяких з ІС, що використовуються на підприємстві. Причому таких сегментів може бути кілька. Стають очевидними проблеми, з якими стикається співробітники підприємства: багаторазове внесення в ІС одних і тих же даних, що найважливіше, не виключається ймовірність помилок при внесенні даних про один і такий же об'єкт. Виходить, що на підприємстві циркулює розрізнена інформація про один об'єкт. Необхідно витрачати час на те щоб «виловлювати» такі записи даних і виправляти їх. Це дуже трудомісткий процес, тому що вона практично ручний і вимагає перегляду величезної обсягу даних. Це – по-перше. А по-друге, кожна інформація має свою актуальність і цінність. Час, до якого дані будуть виправлені, може вплинути на актуальність цих даних.

Управління даними на сучасному підприємстві характеризується наявністю великої кількості різнорідних джерел даних, не пов'язаних єдиними механізмами управління, в тому числі і слабоструктурованих або неструктурованих даних і т.п.

Тому питання інтеграції даних є досить актуальним і затребуваним на даний момент.

Вирішенням подібних проблем є використання таких галузевих рішень як шини даних – (Enterprise Service Bus (ESB)).

Шина Даних – це, в першу чергу, концепція, елемент архітектури IT-ландшафту, який використовується для вирішення завдання інтеграції розрізнених

інформаційних систем в єдиний програмний комплекс з централізованим управлінням передачі інформації і застосуванням сервіс-орієнтованого підходу.

ESB покликана стандартизувати процеси обміну інформацією між внутрішніми системами підприємства, зменшити витрати на додаткову розробку і підтримку цільових систем. Крім того, разом з впровадженим рішенням, отримуємо багаторічний досвід компаній, які розробляли і використовували програмний комплекс тривалий час. Це означає, що більшість основних завдань інтеграції будуть вирішені вже всередині самого продукту і не потребуватимуть додаткових зусиль на аналітику і реалізацію простих рішень.

1 ОГЛЯД ТА АНАЛІЗ ОСНОВНИХ ІНТЕГРАЦІЙНИХ ПЛАТФОРМ

1.1 Огляд інтеграційної платформи WSO2

Платформа інтеграції WSO2 – це широка основа для розробки, повторного використання, запуску та управління інтеграцією. Вона побудована навколо загальної кодової бази інтегрованих технологій з відкритим кодом. Компоненти можна використовувати окремо або як згуртовану інтеграційну платформу.

WSO2 пропонує комплексне рішення, що складається з більш ніж 20 продуктів, що закривають весь спектр завдань від інтеграції і управління API до аналітики та управління правами доступу. Архітектура рішення дозволяє використовувати тільки ті продукти, які необхідні для вирішення конкретних завдань.

Підтримка хмарних технологій закладена в основу архітектури рішень WSO2. Існує два варіанти розгортування компонентів WSO2 – в локальній інфраструктурі, чи на хмарі.

Платформа WSO2 включає в себе:

– WSO2 Enterprise Integrator – потужна інтеграційна шина. Вона забезпечує обмін і перетворення даних між системами, SaaS-додатками, сервісами і API. Рішення включає ESB, брокер повідомлень, сервер додатків, сервер служб даних, сервер виконання бізнес-процесів і платформу для розміщення мікросервісов msf4j. Структура EI представлена на рисунку 1.1.

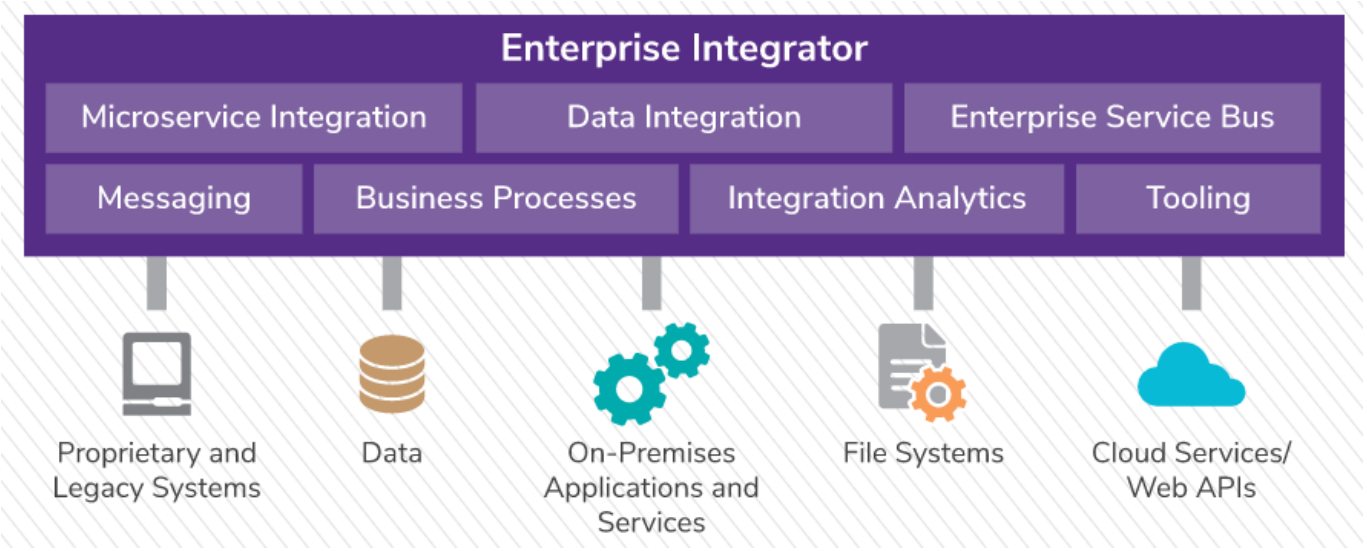

Рис. 1.1 – Основні компоненти WSO2 EI

– WSO2 API Manager (API-M) – програма для управління API, яка дозволяє організаціям створювати, публікувати та керувати API. APIM керує життєвим циклом API, розробку додатків, контролює доступом, обмеження швидкості та аналітика в одній системі. Структура API-M представлена на рисунку 1.2;

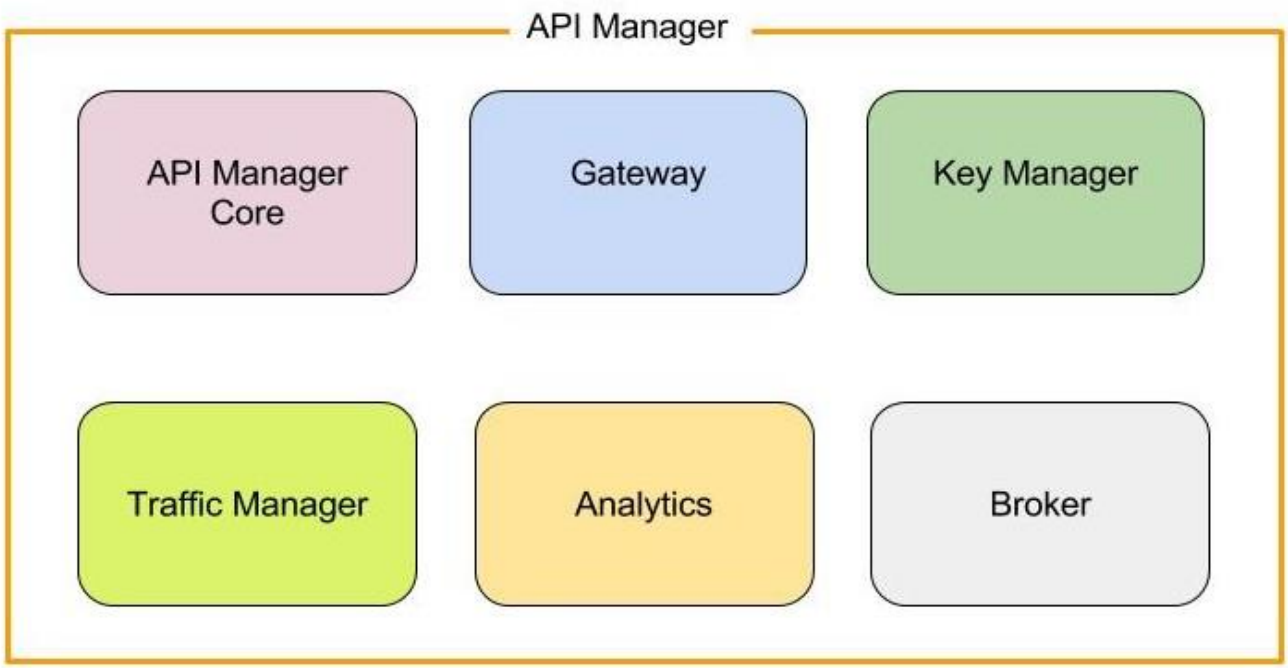

Рис. 1.2 – Основні компоненти WSO2 API-M

– WSO2 Identity Server (IS) – ефективно виконує складну задачу управління ідентифікацією, безпекою та конфіденційністю для корпоративних додатків, сервісів і API;

– WSO2 Stream Processor – потоковий SQL-движок, який підтримує аналітику в реальному часі;

– Ballerina – мова програмування, основними відмінностями якої являються паралельність і жорстка типізація, із текстовими та графічними синтаксисами, оптимізована для інтеграції;

– WSO2 Data Analytics Server – сервер пропонує інтелектуальний і високоефективний аналіз продуктових даних в реальному часі, дозволяє обробляти дані, ідентифікувати шаблони і реагувати протягом мілісекунд;

– WSO2 IoT Server – дозволяє підключатися і керувати своїми пристроями. Можна керувати програмами, налаштуваннями безпеки і даними, передбачена візуалізація даних.

Всі продукти WSO2 поширюються під ліцензією з повністю відкритим вихідним кодом. Це дозволяє використовувати продукти без будь-яких ліцензійних платежів. Можна виконати модифікацію будь-яких елементів.

Всі продукти WSO2 побудовані на одній платформі – Carbon, що забезпечує наскрізну інтеграцію і єдиний підхід до розгортання і управління всіма підсистемами комплексу.

1.2 Огляд інтеграційної платформи IBM

IBM Cloud має багату історію інтеграції, і компанія базується на цьому досвіді щоб розробити сучасну інтеграційну платформу, просту в забезпеченні та розгортанні в гібридних та багатошарових середовищах.

IBM Cloud Pak for Integration охоплює всі аспекти інтеграції, включаючи обмін повідомленнями, трансляцію подій та швидкісну передачу даних. Він пропонує єдиний логін для доступу до декількох інструментів та управління доступністю. Платформа охоплює гнучку стратегію інтеграції, що дозволяє вам вибрати потрібну

форму інтеграції, будь то її мікросервіси, контейнеризація, обмін повідомленнями, API чи інтеграція додатків. І нарешті, щоб підтримати динамічні потреби у мультихмарних середовищах, платформа може швидко масштабуватися.

Так як і її «опенсоурс» конкурент WSO2, платформа пропонує такі свої компоненти:

– IBM API Connect – створення, захист та керування всією екосистемою API в хмарних середовищах;

– IBM App Connect (ESB) – інтеграційне рішення із сотнями вбудованих конекторів для локальної системи передачі даних та SaaS. Приклад взаємодії App Connect з різними джерелами даних показано на рисунку 1.3;

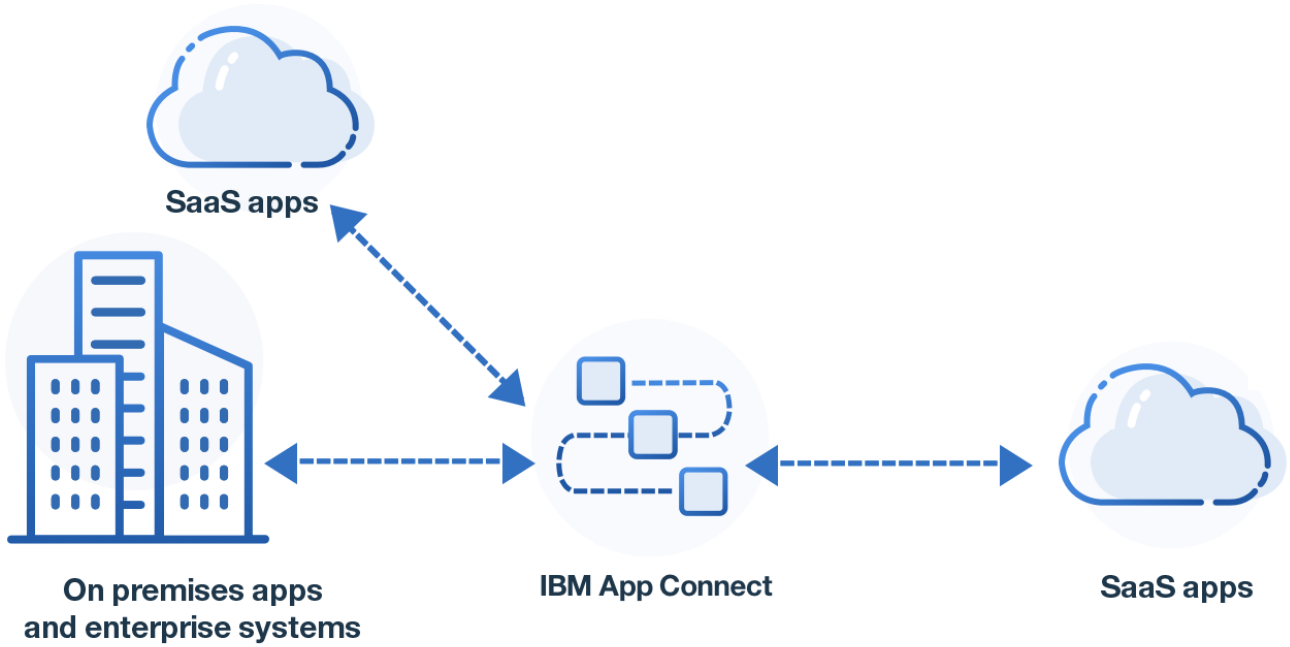

Рис. 1.3 – взаємодія App Connect з різними джерелами даних

– IBM MQ (WebSphere MQ) – надає універсальну мережу обміну повідомленнями з високою сумісністю з іншими програмами, за допомогою якої можлива швидка і надійна відправлення повідомлень для додатків, а також уможливлює інтеграцію архітектури на основі служб (SOA) і існуючих IT-ресурсів;

– IBM DataPower Gateway (IBM DPG) – полегшує виконання вимог організацій в області захисту та інтеграції цифрових бізнес-процесів за рахунок

єдиного багатоканального шлюзу. Це рішення допомагає забезпечити безпеку, контроль і інтеграцію, а також надає оптимізований доступ до повного спектру робочих завдань, пов'язаних з мобільними і хмарними середовищами, веб-додатками, інтерфейсами програмування додатків (API), сервіс-орієнтованою архітектурою (SOA) і B2B. Місце IBM DPG в інтеграційній інфраструктурі показано на рисунку 1.4;

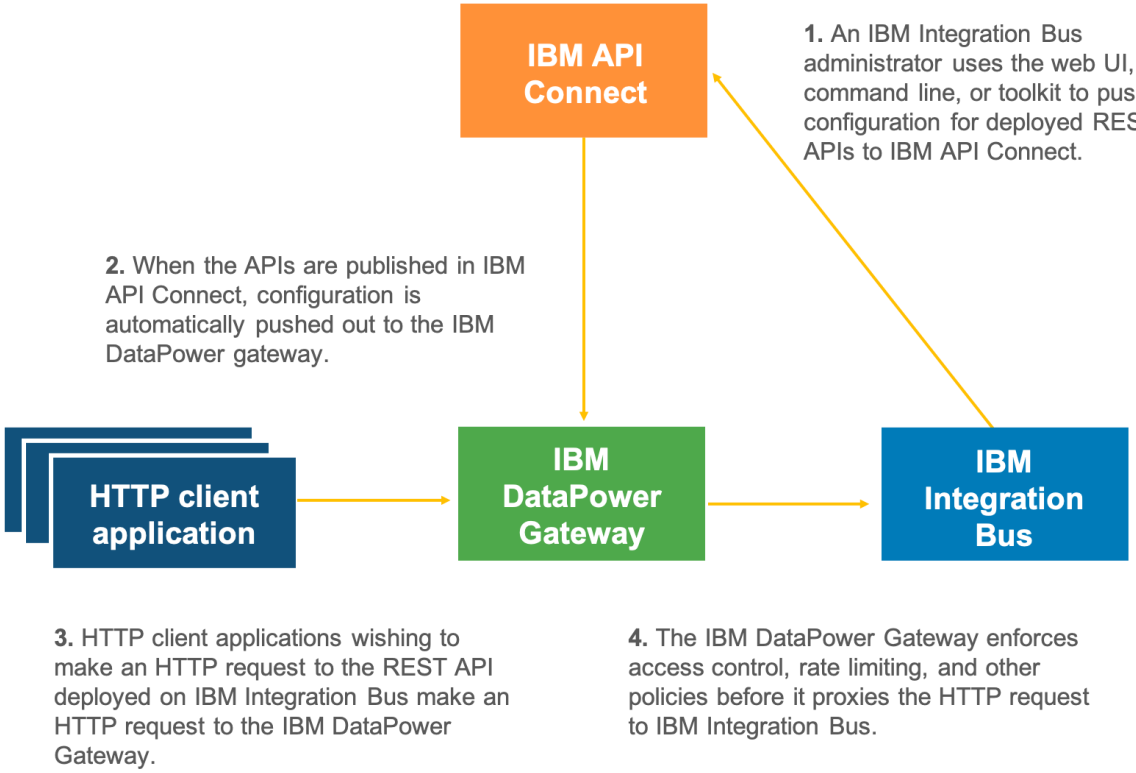

Рис. 1.4 – Приклад взаємодії IBM DPG з іншими компонентами

– IBM Secure Gateway Service – забезпечує швидке, просте та безпечне рішення для підключення будь-чого до будь-чого. Рішення забезпечує стійкий зв’язок між локальними або сторонніми хмарними середовищами та IBM Cloud;

– IBM Event Streams – це високопродуктивна відмовостійка платформа потокової обробки подій на базі Apache Kafka для додатків, керованих подіями;

– IBM Aspera – дозволяє передавати файли по всьому світу на максимальній швидкості незалежно від розміру, відстані і стану мережі. Запатентована технологія FASP Technology (Fast, Adaptive, Secure, Protocol – швидкий, адаптивний, захищений, протокол) є випробуваним стандартом високошвидкісної передачі великих файлів по глобальних мережах (WAN);

IBM App Connect Enterprise v11 поєднує існуючу довірену галузеву технологію IBM Integration Bus з новими хмарними технологіями та IBM App Connect Professional, щоб забезпечити платформу, яка підтримує повну ширину потреб інтеграції в сучасному цифровому підприємстві.

1.3 Огляд Mule ESB

Mule ESB – це легка інтеграційна платформа (ESB), яка дозволяє розробнику об'єднувати різні інформаційні системи на основі принципів обміну повідомленнями (message routing), зіставлення даних (data mapping), управління повідомленнями (orchestration), надійності (контроль за обміном повідомленнями), захисту (використання https і опціональних конекторів) і масштабування між вузлами (коннекторами).

Архітектура являє собою масштабований, розподілений об'єкт-брокер, який може легко керувати взаємодіями між додатками різних виробників, включаючи хмарні і з використанням майже всіх сучасних протоколів.

Mule ESB як елементи обробки повідомлень (трансформаторів) може використовувати вставки коду на популярних мовах програмування (Java, Groovy, Ruby, JavaScript, Python). Тексти програм Mule ESB написані на Java (платформа Java EE) і відповідно підтримується взаємодія з даними стеком технологій (готовий додаток може бути запущено на сервері додатків Apache Tomcat).

Mule ESB побудована на принципі обміну повідомленнями між коннекторами - об'єкт "MuleMessage" містить в собі об'єкт "Payload" – корисне навантаження повідомлення. Шляхом трансформації і маршрутизації повідомлень можна створити необхідний інтеграційний процес (flow). Mule ESB дозволяє створювати інтеграційні процеси (flow) з використанням патернів (pattern - не мають графічного відображення в MuleStudio) або безпосередньо flow (flow конструюється шляхом підключення компонентів з потрібних палітр MuleStudio). Flow Mule ESB є XML схемами. Додаток може містити кілька flow для вирішення різних завдань.

Можливості даної інтеграційної платформи:

– дозволяє зв'язуватися з великою кількістю протоколів: SOAP, REST, JMS, MQ, JBI, AQ, Caching, JavaSpaces, GigaSpaces, Email, IM, JCA, AS400 Data Queues, System I / O;

– інтеграція додатків або систем безпосередньо або з використанням хмарних конекторів;

– використання конекторів "з коробки" для інтеграції SaaS додатків;

– створення і експонування (надання в публічний доступ) API;

– використання готових API;

– створення веб-сервісів керуючих повідомленнями від інших веб-сервісів;

– створення інтерфейсів для експонування додатків (надання в публічний доступ);

– інтеграція B2B з рішеннями, які просто захищати, мають високу ефективність і прості в розгортанні;

– експортування додатків в хмарні сервіси.

Для розробника надається інструментарій "Mule Studio" – середовище розробки заснована на популярної IDE (інтегрованому середовищі розробки) Eclipse, дозволяє створювати, запускати і налагоджувати Mule проекти.

Mule ESB є відкритим програмним забезпеченням (CPAL-ліцензія). Назва Mule (Мул) було дано, так як Mule ESB бере на себе велику частину розробницького навантаження (полегшує працю розробника інтеграційної системи і забезпечує належну продуктивність).

Платформа орієнтована на Java, але може бути брокером для інших платформ, таких, як .NET за допомогою веб-служб або сокетів.

1.4 Порівняння платформ за їх можливостями

Для порівняння я вибрав дві інтеграційні платформи WSO2 і IBM, ось основні їх переваги і недоліки.

1.4.1 Огляд переваг і недоліків платформи WSO2

– Переваги:

– відкрита кодова база продукту;

– всі компоненти платформи написані на Java, це дає можливість користувачу системи писати свої Java-класи і використовувати їх;

– хмарна PaaS;

– всі продукти, ліцензовані на програмне забезпечення Apache;

– не тільки бібліотеки, але і веб-GUI, DEV-Tools та репозиторії OSGi-P2.

– відповідає стандартам WS;

– швидкий старт для нових користувачів з великою кількістю готових прикладів;

Складнощі:

– дуже активна розробка WSO2, нелегко слідувати за новими версіями продукту;

– деякі компоненти знаходяться на ранній стадії

– адміністрування системи(якщи не використовувати хмарною версією платформи)

Актуальна версія Enterprise Integrator – 7.0 це повна API-орієнтована гібридна інтеграційна платформа, яка забезпечує:

– вибір архітектурних підходів як для централізованих, так і для децентралізованих (розподілених по мережі) методів;

– вибір стилю інтеграції заснованих на кодовій базі, так і на графічних конфігураціях;

– архітектура, оптимізована для хмарних інтеграцій та безпроблемний підхід до розгортання мікросервісів.

WSO2 EI 7.0 розглядає популярні підходи до інтеграції з трьома індивідуальними компонентами:

– Ballerina Integrator – революційний, кодовий підхід для гнучкої інтеграції. Він заснований на мові Ballerina для програмування мережево розподілених програм;

– Micro Integrator – включає широко поширений механізм інтеграції з відкритим кодом для централізованого стилю ESB або інтеграції в хмару. Він пропонує інтуїтивно зрозумілий, графічний дизайнер потоків інтеграції;

– Streaming Integrator – заснований на Siddhi, це хмарний, легкий потоковий інтегратор, який розуміє потокові SQL запити для захвату, аналізу, обробки події в режимі реального часу.

1.4.2 Огляд переваг і недоліків платформи IBM

IBM безумовно, більш зрілий продукт, який являється лідером ринку в сфері інтеграційних платформ, а темпи зростання продовжуються

Переваги:

– хмарна PaaS;

– онлайн підтримка по технічним питанням;

– велика кількість конекторів до інших популярних сервісів;

– веб-GUI, DEV-Tools;

– “user-friendly” мапінг даних

Складнощі:

– більшість продуктів IBM платні та потребують ліцензії, IBM App Connect не є виключенням;

– не тривіальне розгортання і керування системи, що в більшості випадків потребує необхідініть в адміністраторі системи;

– власна мова програмування інтеграційних потоків – ESQL.

Актуальна версія ESB – IBM App Connect

IBM App Connect в зв’язці з іншими продуктами IBM такими як API Connect, IBM DataPower Gateway, IBM Aspera та ін. створюють гнучку, швидку та захищену

інтеграційну інфраструктуру, але всі вони потребують ліцензування і є дуже дорогими.

2 ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ WSO2 EI

2.1 Архітектура WSO2 Enterprise Integrator

Основними компонентами WSO2 EI є Ballerina Integrator, Micro Integrator і Streaming Integrator.. Ці компоненти доступні як окремі завантаження. Компоненти можуть легко інтегруватися з сторонніми брокерами (такими як ActiveMQ, RabbitMQ, Kafka та NATS), рішеннями робочого процесу (Camunda) та інструментами спостереження (такими як ELK, Prometheus та Jaeger / Zipkin) для забезпечення розширених функціональних можливостей. Архітектура представлена на рисунку 2.1, а візуальний вигляд програмного коду чи інтеграційного потоку як плагіни в різних IDE для кожного із компонентів представлені на рисунках 2.2-2.4.

Рис. 2.1 – Архітектура WSO2 Enterprise Integrator

2.2 Введення в WSO2 EI

Інтеграційні команди часто застосовують різні архітектурні стилі для інтеграції, використовуючи централізований ESB або більш розповсюджений хмарний підхід, заснований на мікросервісах. Поточні інструменти та підходи до інтеграції часто примушують команди до одного чи іншого підходу затрудняючи різним IT-групам домовлятися про одну технологію, постачальника чи організацію. Більше того, звичайні інтеграційні технології не є ні гнучкими, ні «developer-friendly». З іншого

боку, головна мета мови програмування та фреймворків є зручними для розробників, але надзвичайно складними для використання в якості інтеграційної технології. Такий розподіл між підходами до розвитку та інтеграції сповільнює час виходу на ринок дедалі більшої кількості IT-проектів, які залежать від інтеграції. Це серце «інтеграційного розриву».

Щоб вирішити цей інтеграційний розрив, WSO2 розробив Enterprise Integrator 7.0 (EI 7.0), який є платформою з відкритим кодом, хмарною та гібридною платформою інтеграції для API, потоків даних та потоків подій. Графічна конфігурація та інтеграція, керована кодом – це два підходи для створення інтеграцій, і EI 7.0 – краще рішення для підтримки обох, плюс вибір стилів для децентралізованих хмарних мікросервісів або централізованих архітектурних стилів ESB. Крім того, рішення є основним компонентом WSO2 Integration Agile Platform, яка також включає управління API, ідентифікацію та можливості управління доступом.

2.3 Ключові компоненти

Для створення нашої системи інтеграції необхідно обрати один з трьох основних компонентів – Ballerina Integrator, Micro Integrator чи Streaming Integrator.

2.3.1 Ballerina Integrator

Ballerina Integrator підходить для розробників, які використовують децентралізовану інтеграційну архітектуру, мікросервіси та хмарні програми, з його допомогою можна створити інтеграційні мікросервіси, використовуючи стиль інтеграції «code-first».

Ballerina Integrator заснований на мові програмування Ballerina (ballerina.io), яка призначена для інтеграції та включає в себе першокласні, хмарні функції, а також інструмент графічної діаграми послідовностей. Балерина включає в себе основні концепції інтеграції розподіленої системи в мову та пропонує безпечну типізацію та для «concurrency» середовище для впровадження мікросервісів з розподіленими

транзакціями та гарантованою доставкою повідомлень. Базуючись на взаємодії діаграм послідовності, балерина має вбудовану підтримку загальних інтеграційних моделей та конекторів, включаючи розподілені транзакції, компенсації та вимикачі. Завдяки першокласній підтримці JSON та XML, ballerina поставляється з ключовими мовними конструкціями, такими як кінцеві точки сервісів, типи даних, комунікатори та робітники, що дозволяє просто та ефективно будувати надійні інтеграції через кінцеві точки мережі. Візуальний вигляд програмного коду чи інтеграційного потоку як плагін для Visual Studio Code представлено на рисунку 2.2.

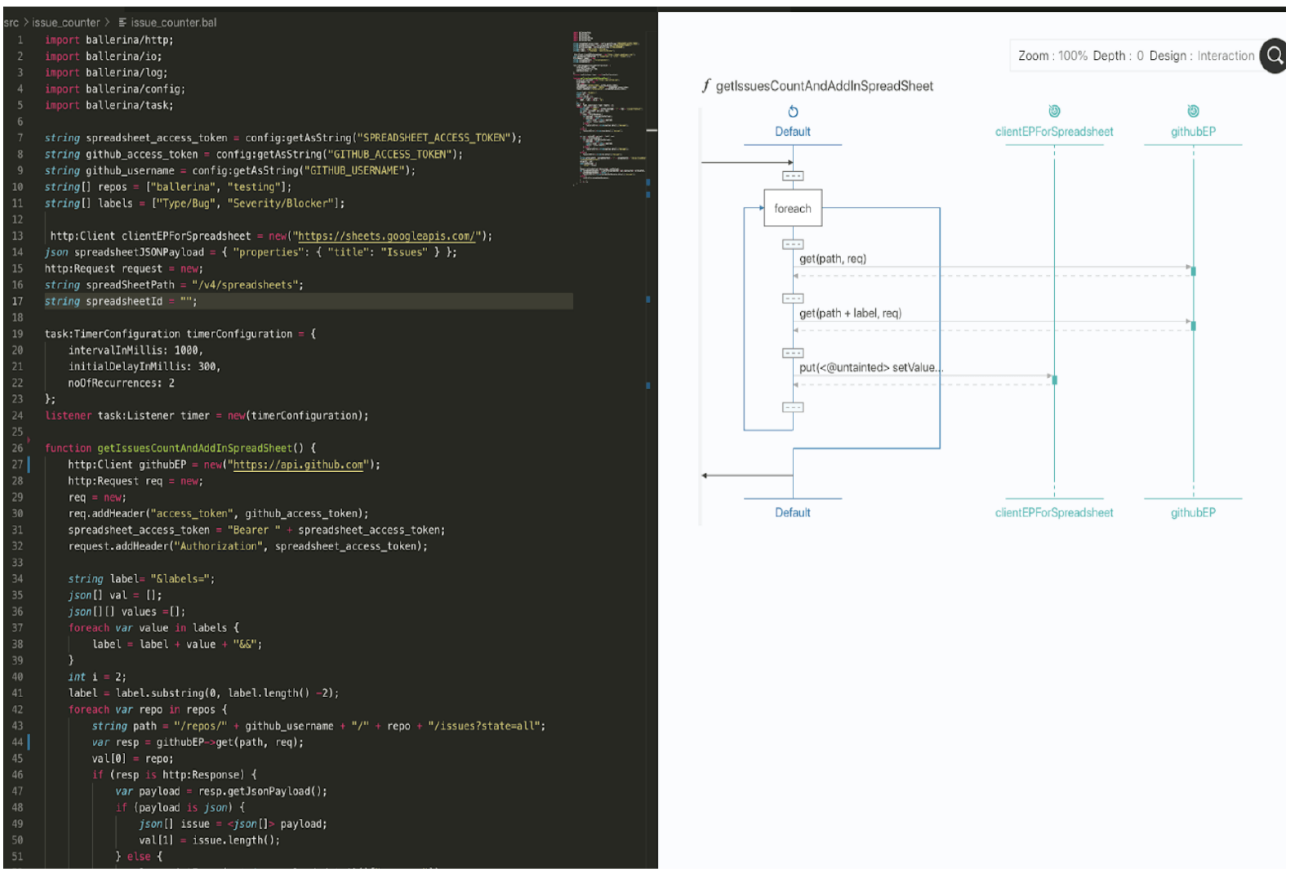

Рис. 2.2 – Ballerina Plugin для Visual Studio Code

Balerina Integrator можна використовувати для створення простих інтеграцій, керованих кодом. Насправді, розробники можуть безпосередньо програмувати інтеграційні проекти, допомагаючи об'єднати сфери розрозбки та інтеграції, тобто команди з кодування та інтеграції по принципу «waterfall».

Ballerina Integrator - це більше, ніж мова програмування і включає в себе інсталятор Ballerina, інструмент візуалізації діаграм послідовностей, ключові додатки балерин, заздалегідь визначені шаблони інтеграції, конектори протоколів (наприклад, HTTP, File, gRPC, AMQP, NATS, Kafka та JMS), конектори додатків (такі як Salesforce, Amazon S3, Amazon SQS і SAP) та розширення для оптимізації балерини для інтеграційних проектів.

2.3.2 Micro Integrator

Micro Integrator підходить для розробників, які дотримуються традиційної централізованої ESB-інтеграції, або децентралізованої інтеграції, щоб реалізувати архітектуру мікропослуг за допомогою підходу, орієнтованого на мінімальний код та графічну конфігурацію. У централізованому підході ESB Micro Integrator може використовуватися як централізований рівень ESB для інтеграції API, даних, подій, потоків та систем. В децентралізованому підході, оскільки Micro Integrator - це легковісна інтеграція в рантаймі і є основним варіантом хмари WSO2 ESB з відкритим кодом, він може розміщувати один або кілька мікросервісів інтеграції та працювати всередині контейнера Docker. Micro Integrator пропонує інструмент дизайнера графічного потоку даних разом з мовою конфігурації на основі XML для створення інтеграцій. Візуальний вигляд інтеграційного потоку як плагін в Eclipse IDE представлено на рисунку 2.3.

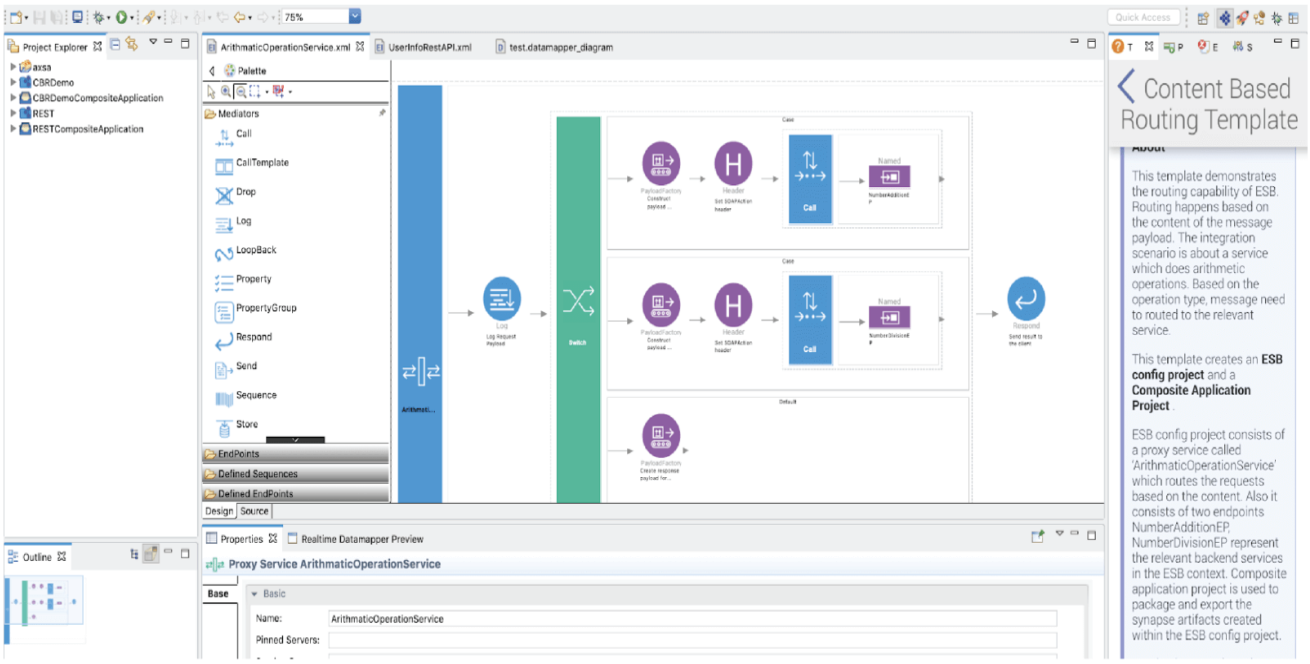

Рис. 2.3 – Integration Studio plugin в IDE Eclipse для Micro Integrator

Micro Integrator підтримує всі функції ESB WSO2, зокрема широкий спектр транспорту та протоколів та наявність 160+ конекторів та адаптерів для різних категорій, таких як платежі, CRM, ERP, соціальні мережі чи «legacy» системи. Він може легко підключатися та працюватиз різними сховищами даних, такі як RDBMS, CSV, Excel, ODS, Cassandra, Google Spreadsheets, RDF та будь-яка веб-сторінка як сервіс даних. Micro Integrator забезпечує підтримку всіх інтеграційних шаблонів підприємств (EIP), інтеграції баз даних, публікації подій, ведення журналу логів та аудиту, валідації, маршрутизації та перетворення даних. Співставлення даних можна здійснити візуально за допомогою візуального «data mapper», який може перетворити вхідні дані у вихідні дані. Навіть з декларативною розробкою з конфігурацією XML, Micro Integrator також підтримує обробку помилок, відстеження та налагодження логіки посередництва повідомлень. Мова конфігурації може бути розширена за допомогою користувальницьких DSL за допомогою шаблонів, а розробники можуть використовувати плагін Integration Studio, який постачається з багатими графічними редакторами та іншими стандартними інструментами, для безшовної конфігурації логіки інтеграції.

2.3.3 Streaming Integrator

Командам інтеграції неминуче потрібно інтегрувати події та потокове передавання даних, часто вимагаючи складної обробки та ініціювання подій, або ще гірше, якщо мають справу з необробленими даними. Більшість процесів обробки та запуску важко налаштувати, а потім з'єднати з базовою інтеграційною платформою, наприклад, потоки несуть великий обсяг даних, що рухаються з високою швидкістю, а джерела / кінцеві точки даних можуть бути різними за протоколами, форматами даних тощо.
Окрім Ballerina Integrator та Micro Integrator, EI 7.0 пропонує третій інтегратор, відомий як Streaming Integrator. Потоковий інтегратор ідеально підходить для систем, які мають обробляти та діяти над потоковими даними. Це власний хмара і легкий механізм обробки потоків на основі Siddhi, який розуміє потокові SQL запити для збору, аналізу, обробки та дії на події в режимі реального часу. Це полегшує інтеграцію потокових даних у реальному часі та аналітику та може використовуватися як в монолітній, так і в розподіленій архітектурі. Потоковий інтегратор підтримує повну інтеграцію з Micro Integrator і повністю сумісний з основними системами потокового обміну повідомленнями, такими як Kafka та NATS. Він може витягувати дані з основних баз даних і навіть може інтегрувати дані у хмарне сховище. Візуальний вигляд інтеграційного потоку представлено на рисунку 2.4

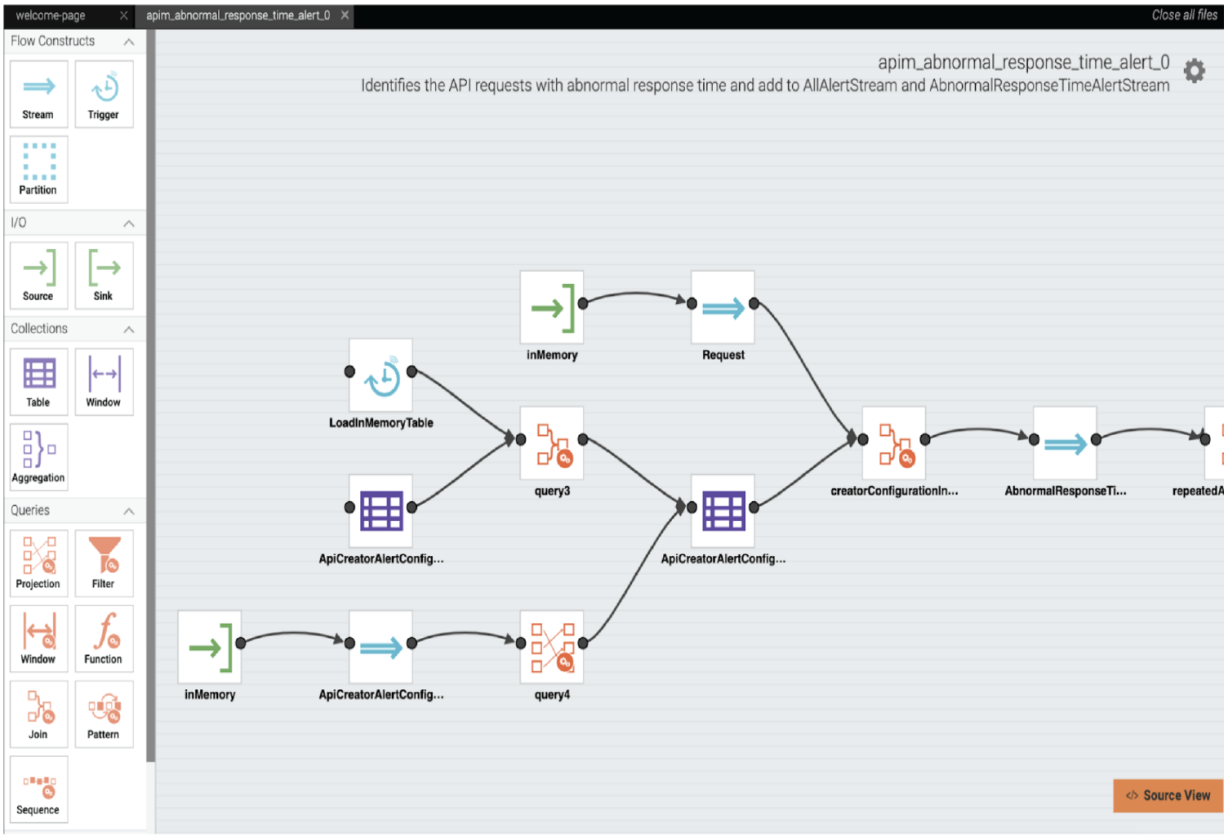

Рис. 2.4 – Streaming Integrator tooling

2.3.4 Data Services

Дані у організації можуть бути складним набором інформації, яка зберігається в неоднорідних системах, починаючи від RDBMS-файлів до файлів Excel та електронних таблиць Google тощо. Сервіси передачі даних створюються з метою отримання даних із її інфраструктури. Іншими словами, коли створюємо службу передачі даних в WSO2 Micro Integrator, дані, які зберігаються в системі зберігання даних (наприклад, RDBMS), можуть бути придставлені у вигляді сервісу. Це дозволяє користувачам (це може бути будь-яка програма або система) отримати доступ до даних без взаємодії з вихідним джерелом даних. Таким чином, сервіси передачі даних є зручним інтерфейсом для взаємодії з рівнем бази даних в організації.

Сервіс даних у WSO2 Micro Integrator – це веб-сервіс на основі SOAP за замовчуванням. Однак також є можливість створити ресурси REST, що дозволяє

програмам та системам, що споживають послугу даних, мати доступ як до SOAP, так і до RESTful доступу до даних.

Дані організації можуть зберігатися в різних системах зберігання даних, які називаються джерелами даних. Підтримуються такі джерела даних:

- реляційні бази даних;
- файли CSV;
- таблиці Microsoft Excel;
- електронні таблиці Google;
- RDF;
- MongoDB, Cassandra;
- веб-ресурси;
- JNDI datasources.

Крім того, можна використовувати та створювати власні джерела даних.

Сервіс даних в WSO2 Micro Integrator, має можливість агрегувати дані, що зберігаються в різних, розрізнених джерелах даних та представляти дані як єдиний вихід. Наприклад, дані працівників у компанії можуть зберігатися в різних сховищах даних (подробиці трудової історії, реквізити офісу, контактна інформація тощо).

Об’єднання даних дозволяє користувачам споживати всі ці дані через один запит до служби даних. Сервіс даних збирає відповідні дані з різних джерел та подає їх як одну відповідь на запит.

2.3.5 WSO2 Identity and Access Management

Управління ідентичністю та доступом (IAM) – це ефективна інтеграція та управління ідентичностями, що надає користувачам доступ до потрібних ресурсів у потрібний час. Ідентичність – це вже не просто проект безпеки для підприємств. Що стосується інтеграції та домену API, оскільки бізнес продовжує збільшувати кількість внутрішніх та сторонніх API, важливіше, ніж будь-коли, бути впевненим, що API інтегруються та керуються безпечно. В домені користувача із збільшенням простору ідентичності користувачів, політик в масштабах компанії, складної структури ієрархії

та ролей та програм орієнтованих на клієнтів, безпека стає щоденним завданням для архітекторів і адміністраторів ідентичності.

Сервер WSO2 Identity є рішенням як для API, так і домену користувача, надаючи розширений досвід користування WSO2 Integration Agile Platform. Це надзвичайно розширюваний продукт IAM, розроблений для захисту API та мікропослуг та забезпечення можливості ідентичності клієнта та управління доступом (CIAM). Функціональна діаграма IAM представлена на рисунку 2.5.

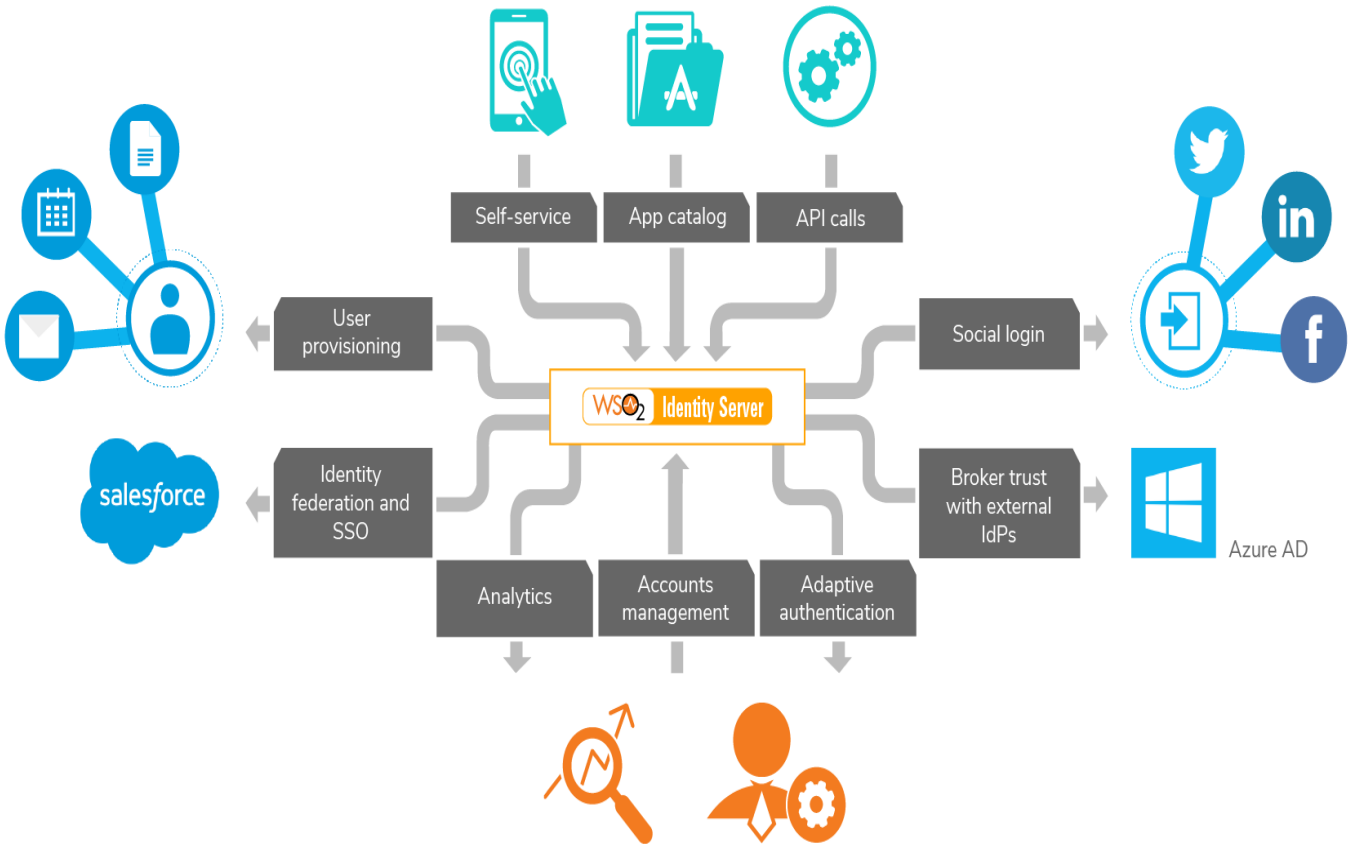

Рис. 2.5 – Функціональна діаграма IAM

2.3.6 Інші сервіси

Інтеграційна платформа WSO2 включає в себе велику кількість продуктів та сервісів і всі вони є «open-source». Ще декілька основних компонентів які не були описані детально:

– сервер бізнес-процесів(BPS) – управління тривалими бізнес-процесами;

– Message Broker – для посередництва та надійного обміну повідомленнями;

– MSF4J – надає можливість створення мікросервісів на Java для полегшення складних випадків використання інтеграції;

– сервер аналітики – надає можливість моніторингу статистики інтеграційних потоків (обробляються під час виконання ESB), а також бізнес-процесів (обробляються під час виконання бізнес-процесів);

– API Manager – створення та керування життєвим циклом всіх API організації.

3 РОЗРОБКА СТРУКТУРНОЇ ТА ФУНЦІОНАЛЬНОЇ СХЕМИ СИСТЕМИ ІНТЕГРАЦІЇ ГЕТЕРОГЕННИХ ДЖЕРЕЛ ДАНИХ

3.1 Розробка структурної схеми системи інтеграції гетерогенних джерел даних

Структурна схема системи інтеграції гетерогенних джерел даних зображена на рисунку 3.1 та в Додатку А.

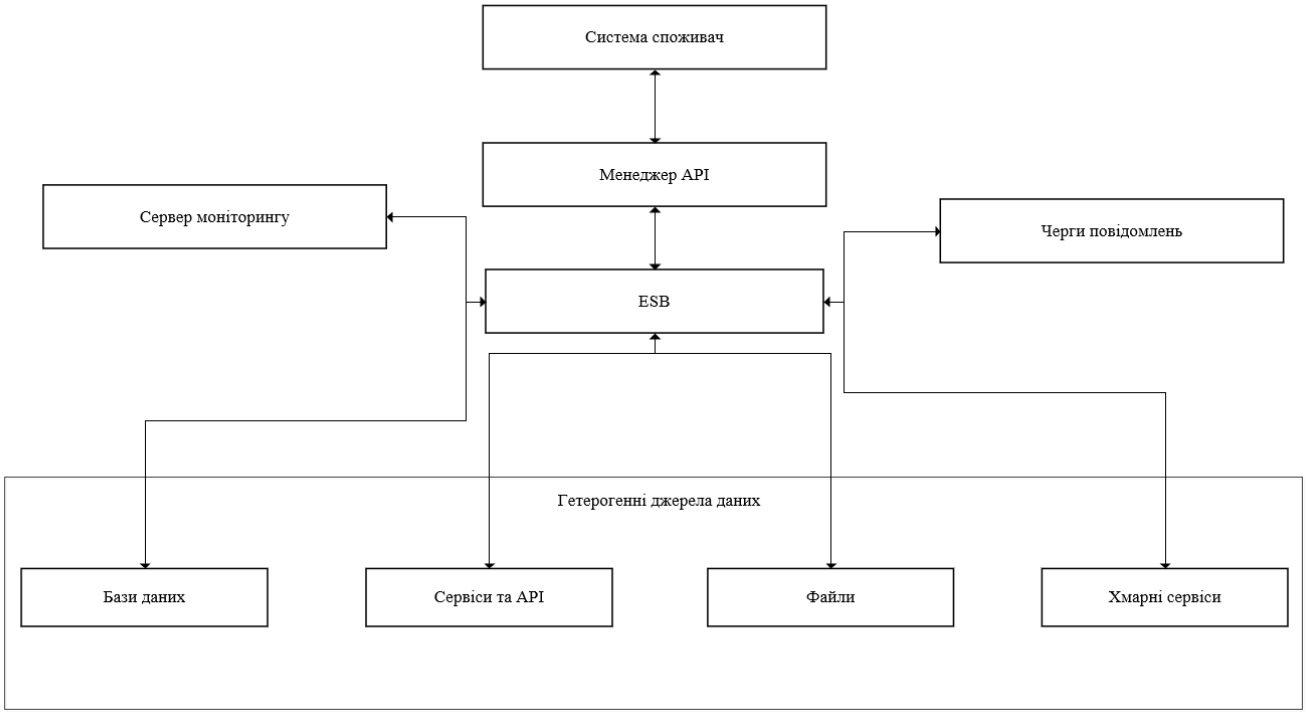

Рисунок 3.1 – Структурна схема системи інтеграції гетерогенних джерел даних

Структурна схема системи інтеграції гетерогенних джерел складається з системи споживача, API менеджера, серверу моніторинга, черг повідомлень, шини даних (ESB) та гетерогенних джерел даних.

3.1.1 Система споживач

Система споживач – це будь-яка система, сервер, чи додаток, що виконує запит до на нашої системи інтеграції. Так як точкою доступу до нашої системи є API, то споживачом може виступати будь-що, що зможе створити http запит.

Всі запити від системи споживача оброблюються менеджером API, і відповідь споживачу поступляє від менеджера API.

3.1.2 Менеджер API

Менеджер API – система, що керує життєвим циклом всіх API, відповідальна за безпечніть запитів, перенаправлення запитів по корректним адресам, валідацію запитів, перевірку прав доступу до запитуваного ресурсу.

Спочатку всі запити ідуть на менеджер API, а потім API-M вирішує до якого конкретного сервісу було адресовано запит і якщо запит пройшов всі необхдні етапи валідації – виконується «redirect» до конкретного сервісу.

Відповідь системі споживачу також проходить через менеджер API, де можуть бути добавлені заголовки до http відповіді чи навіть добавлено щось в тіло відповіді.

3.1.3 ESB

ESB (шина даних) – сервер інтеграції, містить в собі системи інтеграції які працюють по різним протоколам, та в різній формі. Деякі системи мають точку доступу, тобто роботу з ними можна ініціювати самостійно, деякі працюють по трігерам. В системі, що ми розробляємо є http точка входу, до якої запити потрапляють через менеджер API. Сама система, в свою чергу, може виконувати запити до різних джерел інформації для подальшої роботи на над ними. Також ESB може бути інтегрованою з чергою повідомлень.

Щоб мати представлення про кількість, успішність та швидкість обробки запитів підключена система моніторингу. Системою моніторингу виступає компонент продукту WSO2 – Analytics Server.

3.1.4 Гетерогенні джерела даних

Гетерогенні джерела даних – джерела інформації для системи інтеграції, що розгорнена на сервері інтеграції (ESB). Представлено джерела інформації чотирьох типів:

- бази даних;
- сервіси та API;
- файли (FTP сервер);
- хмарні сервіси.

3.2 Розробка функціональної схеми системи інтеграції гетерогенних джерел даних

Функціональна схема системи інтеграції гетерогенних джерел даних зображена на рисунку 3.2 та в Додатку Б.

На функціональній схемі зображені елементи, що виконують певні операції та зв’язки між ними, по яким передаються дані у різному вигляді. Дана схема є більш конкретною, по відношенню до структурної схеми. В цьому розділі описані всі функціональні вузли для побудови системи інтеграції гетерогенних джерел даних.

Розроблювана функціональна схема складається з шістьох основних блоків: системи споживача, менеджера API, серверу моніторинга, шини даних(ESB), черг повідомлень та гетерогенних джерел даних.

Менеджер API відповідає за зв'язок системи споживача(ініціатора) та шини даних, яка виконує всю роботу над отриманою інформацією.

Шина даних складається з таких блоків як:

- транспортний;
- адміністративний (кластерізація / балансування навантаження / доступність);
- message builder (представлення та валідація вхідного повідомлення);
- якісний (гарантована доставка повідомлення/безпека);

– synapse runtime – містить в собі сущності брокерів повідомлень, java-класи медіаторів, сконфігуровані кінцеві точки, сконфігуровані задачі, час виконання яких встановлюється вручну;

– message formatter (трансформація даних в необхідний формат для системи споживача).

В транспортному блоці наведено декілька протоколів, по яким ESB може працювати і обмінюватись інформацією з іншими елементами системи.
Гетерогенними джерелами даних є різні бази даних як SQL так і NoSQL, веб-сервіси, API, SaaS, тобто будь-яка система яка працює по http, https, SOAP протоколах, хмарні сервіси та FTP сервери.

За допомогою існуючих протоколів в транспортному блоці ESB, шина даних має можливість працювати с чергами повідомлень різних типів:

– Message Broker;
– Java Message Service;
– MQ.

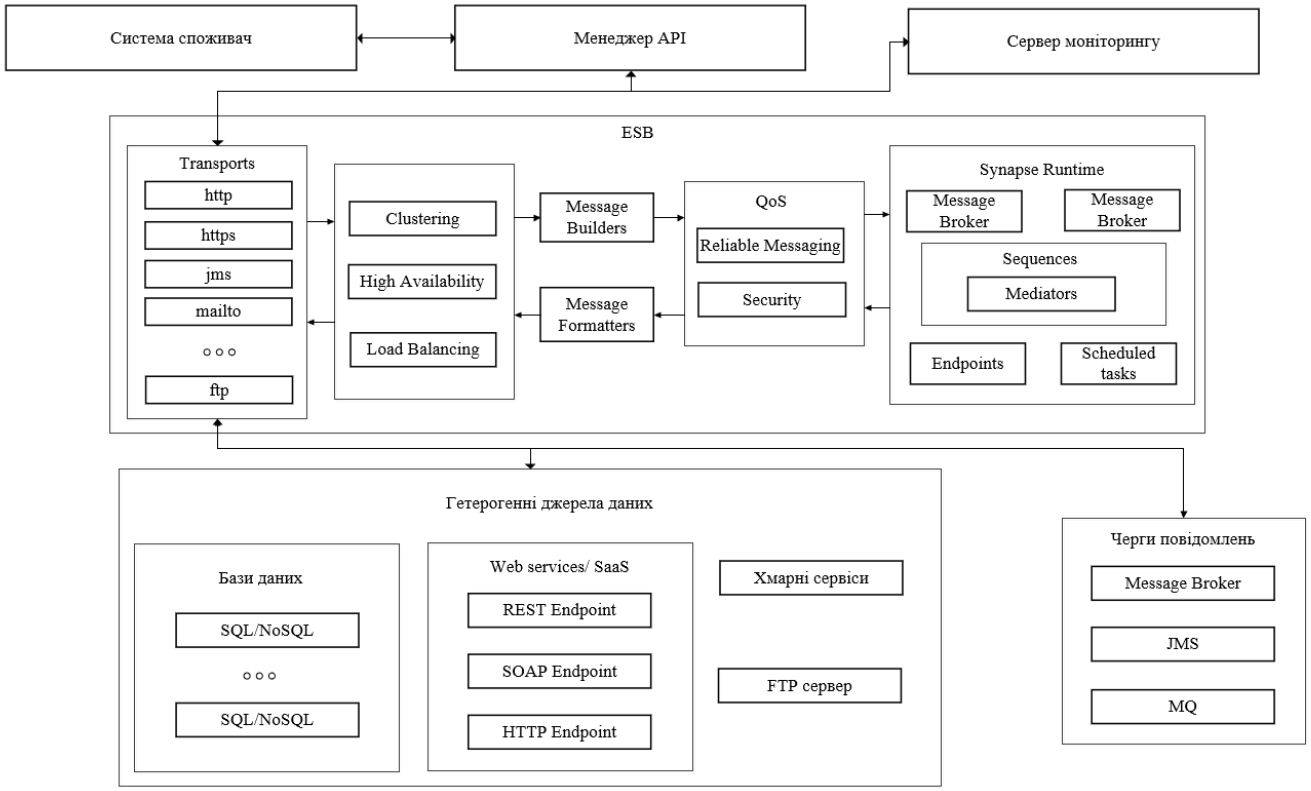

Рисунок 3.2 – Функціональна схема системи інтеграції гетерогенних джерел даних

4 ПОСЛІДОВНІСТЬ СТВОРЕННЯ ТА РОЗГОРТАННЯ СИСТЕМИ

4.1 Системні вимоги та варіанти розгортання WSO2 EI

Micro Integrator продукту WSO2 EI можна встановлювати на фізичній чи віртуальній машині та як контейнер в Docker чи Kubernetes.

4.1.1 Мінімальні вимоги та рекомендації

Micro Integrator встановлюється разом з OpenJDK за замовчуванням, що дозволяє запускати сервіс, як тільки він встановлений. Можна використовувати інший JDK, для цього необхідно вказати змінну середовища JAVA_HOME на новий JDK, переконавшись, що версія JDK сумісна з продуктом WSO2.

Не рекомендується використовувати Apache DS у виробничих умовах через проблеми з масштабованістю. Натомість необхідно використовувати LDAP, на приклад OpenLDAP.

Вимоги для розгорнення в Docker:

– 512 Мб «heap size» для одного екземпляра Micro Integrator. Цього, як правило, достатньо для обробки типових повідомлень формату SOAP. Однак вимоги залежать від розмірів повідомлень та кількості, які одночасно обробляються;

– 1 Гб пам'яті для контейнера Docker;

– 0,5 ядра на екземпляр мікроінтегратора в Docker.

Вимоги для розгорнення на фізичній чи віртуальній машині:

– Мінімум 0,5 ядра(1.0-1.2 GHz Opteron/Xeon processor);

– 1 GB RAM для JVM.

– 512 Мб «heap size» для одного екземпляра Micro Integrator. Цього, як правило, достатньо для обробки типових повідомлень формату SOAP. Однак вимоги залежать від розмірів повідомлень та кількості, які одночасно обробляються.

4.1.2 Розгортання та запуск на фізичній машині чи VM за допомогою інсталятора

За допомогою інсталятора завантаженого з офіційного сайту встановлюємо продукт. Залежно від ОС за замовчуванням WSO2 EI буде встановлено в відповідну директорію:

- Mac OS – /Library/WSO2/EnterpriseIntegrator/7.0.0/micro-integrator;
- Windows – C:\Program Files\WSO2\Enterprise Integrator\7.0.0\micro-integrator;
- Ubuntu – /usr/lib/wso2/wso2ei/7.0.0/micro-integrator;
- CentOS – /usr/lib64/wso2/wso2ei/7.0.0/micro-integrator.

Запуск за допомогою інсталятора для MacOS/Linux/CentOS виконується командою в терміналі «sudo wso2mi»

Логування працює весь час до запуску інтегратора, що зазвичай займає кілька секунд. Чекаємо поки профіль повністю завантажиться, і з’явиться повідомлення, схоже на "WSO2 Carbon started in n seconds.".

У Windows перейдіть до меню Пуск –> Програми –> WSO2 –> Enterprise Integrator. Це відкриє термінал і запустить відповідний профіль.

Можна вручну запустити скрипт запуску продукту з директорії де був встановлений EI, виконавши команду «sudo sh launcher_micro-integrator.sh». Цей скрипт автоматично призначає змінну середовища JAVA_HOME кореневому користувачеві інстанції Micro Integrator.

4.1.3 Розгортання та запуск на фізичній машині чи VM за допомогою дистрибутиву

Перш ніж виконати сценарій запуску продукту, обов'язково встановлюємо змінну середовища JAVA_HOME на машині. Використовуємо JDK, сумісний з WSO2 Enterprise Integrator.

Відкриваємо термінал і переходимо до каталогу MI_HOME/bin/, де MI_HOME – домашня директорія завантаженого дистрибутива.

Виконуємо команду в терміналі:

– MacOS/Linux/CentOS – «sh micro-integrator.sh»;
– Windows – «micro-integrator.bat».

За замовчуванням порт слухача HTTP – 8290, а порт слухача HTTPS – 8253.

4.2 Створення необхідних проектів системи

WSO2 Integration Studio – це графічне середовище розробки WSO2 Enterprise Integrator, що забезпечує ефективну інтеграцію артефактів та прискорює життєві цикли розвитку. Він включає в себе візуальну палітру інструментів, можливість імпортувати готові конектори до палітри інструментів, конструкцій вищого рівня інтеграції та представлення властивостей для складних конфігурацій.

WSO2 Integration Studio дає можливість для створення таких проектів:

– ESB Solution Project (складається з одного або декількох каталогів проектів. Ці каталоги зберігають різні артефакти ESB, які ви створюєте для своєї інтеграційної послідовності);
– ESB Config Project – зберігає артефакти ESB, які використовуються при визначенні потоку посередництва;
– Registry Resource Project – зберігає ресурси реєстру для інтеграційного потоку, пізніше можна використовувати ці артефакти реєстру, коли ви ініціалізуємо свої послідовності посередництва в проекті конфігурації ESB;
– Data Services Project – щоб почати створювати сервіси передачі даних (.dbs файли) для роботи з різними джерелами даних як сервісами;
– Datasource Project – щоб створювати джерела даних, з якими можна працювати через сервіс передачі даних;
– Connector Exporter Project;

– Composite Application Project – дозволяє упакувати всі артефакти (зберігаються в інших проектах ESB) в одну композиційну програму (C-APP). Потім цей C-APP може бути розгорнутий на сервері ESB.

4.2.1 Моделювання процесу системи інтеграції

Перед створенням самої системи необхідно розуміти які продукти, сервіси та зовнішні системи будуть взаємодіяти між собою і по яким протоколам буде відбуватися обмін інформацією. Все це представлено на рисунку 4.1 та в Додатку В.

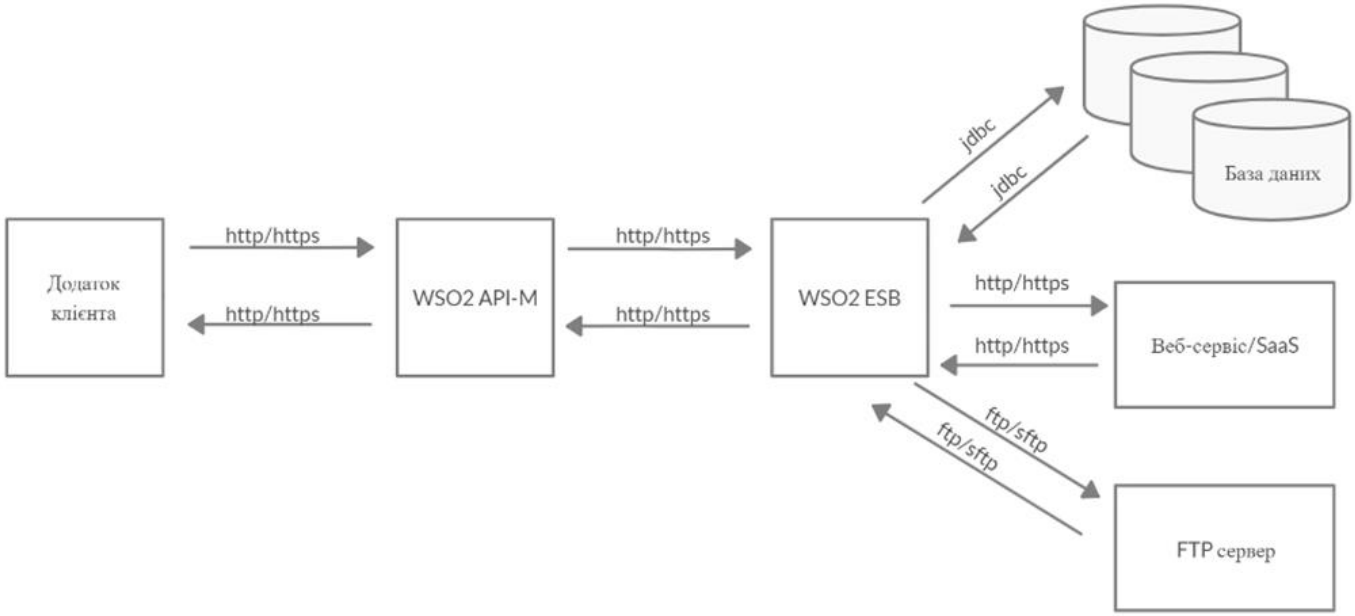

Рис. 4.1 – Модель процесу системи інтеграції

4.2.2 Життєвий цикл створення інтеграційної системи

На рисунку 4.2 та в Додатку Г показана блок-схема, яка описує алгоритм ствроення системи від самого початку і до самого деплою на сервер інтегратора.

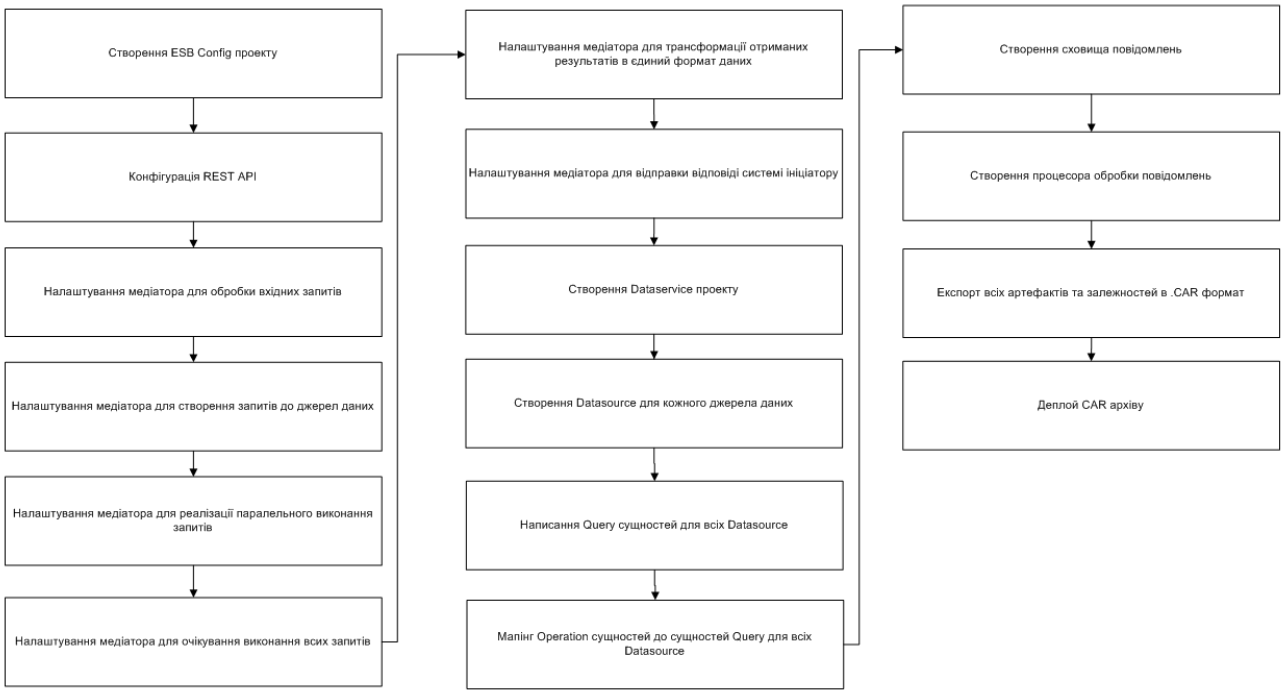

Рис. 4.2 – Блок-схема життєвого циклу створення інтеграційної системи

4.2.3 Створення «ESB Config» проекту

Створення пустого «ESB Config» відбувається в WSO2 Integration Studio.

В WSO2 Integration Studio переходимо на сторінку Miscellaneous, а далі створюємо проект шляхом натиснення кнопки «Create New Config Project». Візуально відображено на рисунку 4.3.

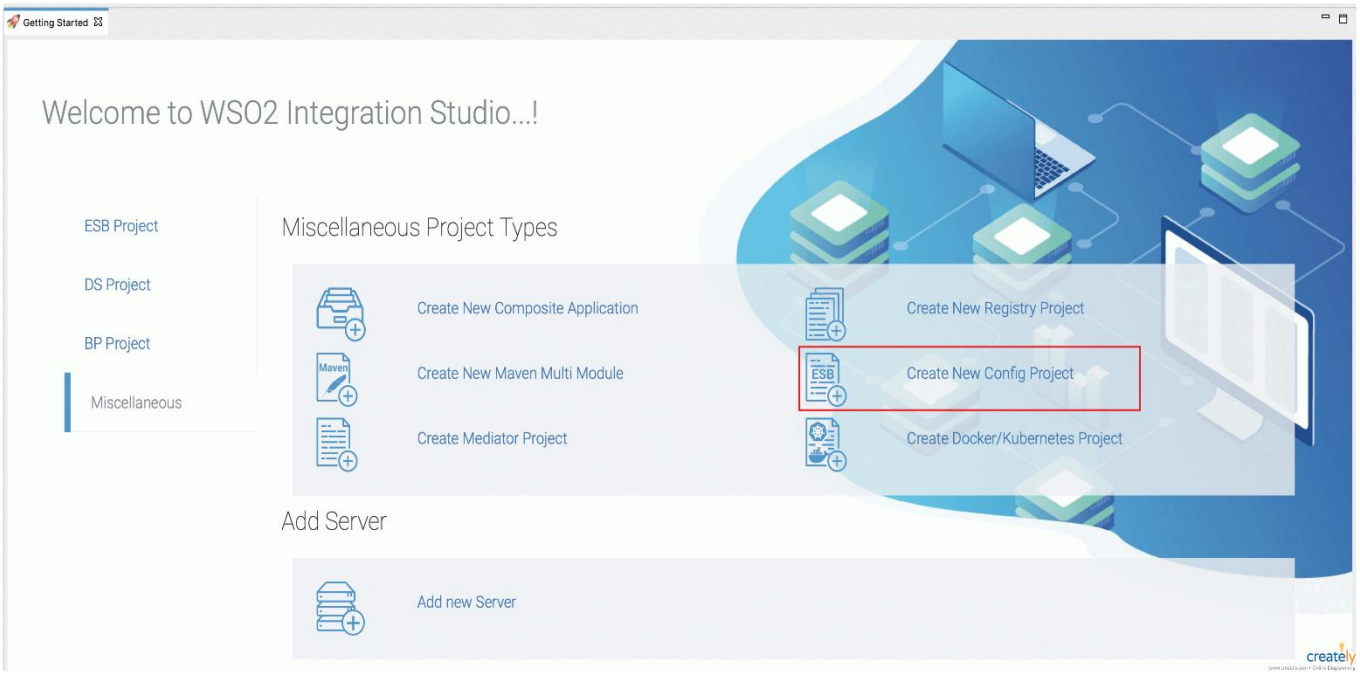

Рис. 4.3 – Створення ESB Config проекту

Після чого даємо імя проекту і можемо переходити до створення артефактів.

Перелік артефактів, що ми можемо створити за допомогою WSO2 Integration Studio:

- REST API;
- Proxy Service;
- Inbound Endpoint;
- Scheduled Tasks;
- Message Store;
- Message Processor;
- Endpoint;
- Endpoint Template;
- Sequence Template;
- Reusable Sequence;
- Registry Resource;
- Local Registry Entries;
- Smooks Configuration;
- Data Service;

– Datasource;
– Data Service Input Validator.

Крім перелічених вище артефактів ми маємо змогу написати свої «custom» артефакти.

4.2.4 Створення Data Service

В WSO2 Integration Studio переходимо на сторінку DS Project, а далі створюємо сервіс шляхом натиснення кнопки «Create New Data Service». Візуально відображено на рисунку 4.4.

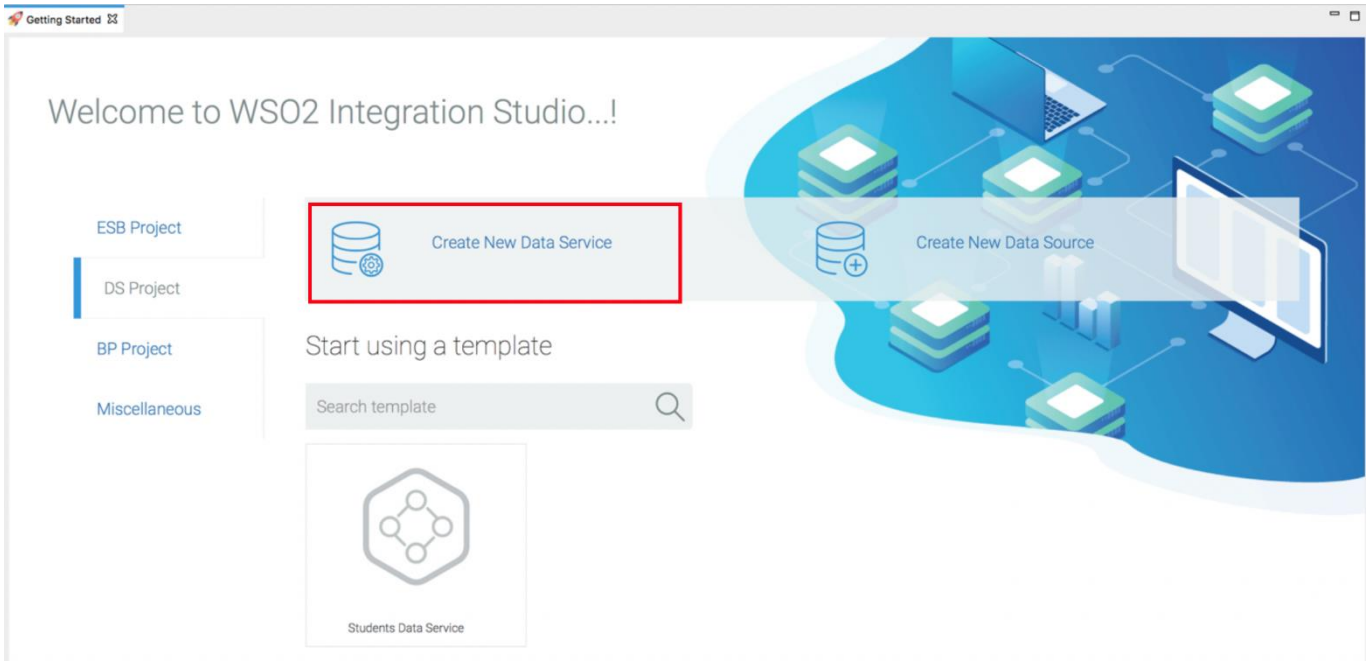

Рис. 4.4 – Створення Data Service

Після цього попадаємо на нову сторінку де натискаємо Next і даємо назву нашому сервісу, продемонстровано на рисунку 4.5.

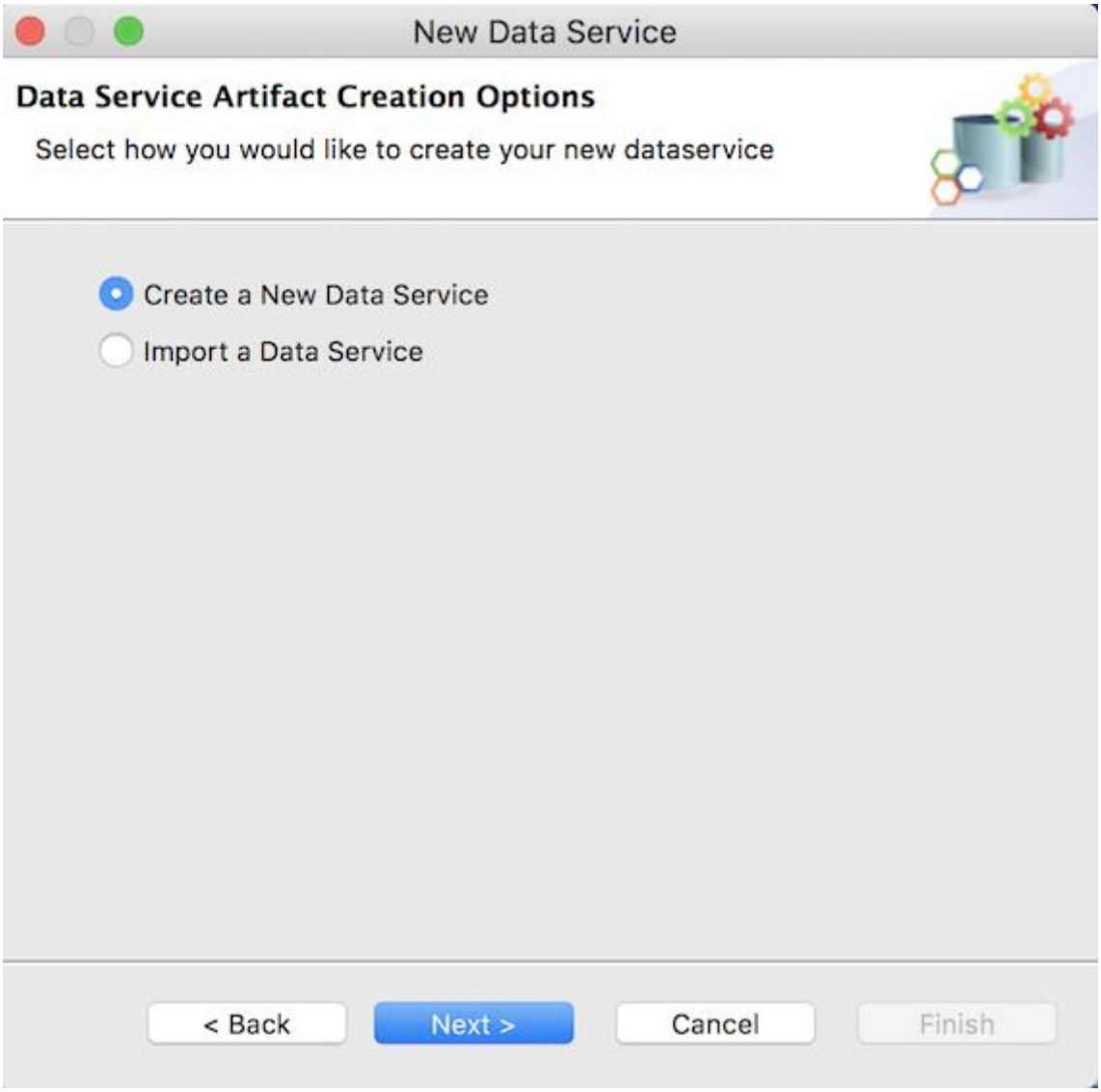

Рис. 4.5 – Створення Data Service

4.2.5 Створення Datasource для нашого Dataservice

Вибираємо вже створений проект Dataservice у навігаторі проектів, натискаємо правою кнопкою миші та переходимо до пункту «New», а потім «Data service».

Відкриється вікно сервісу даних, на рисунку 4.6 показано директорію проекту та вікно конфігурацій самого сервісу, де будуть описуватись всі необхідні datasource проекту.

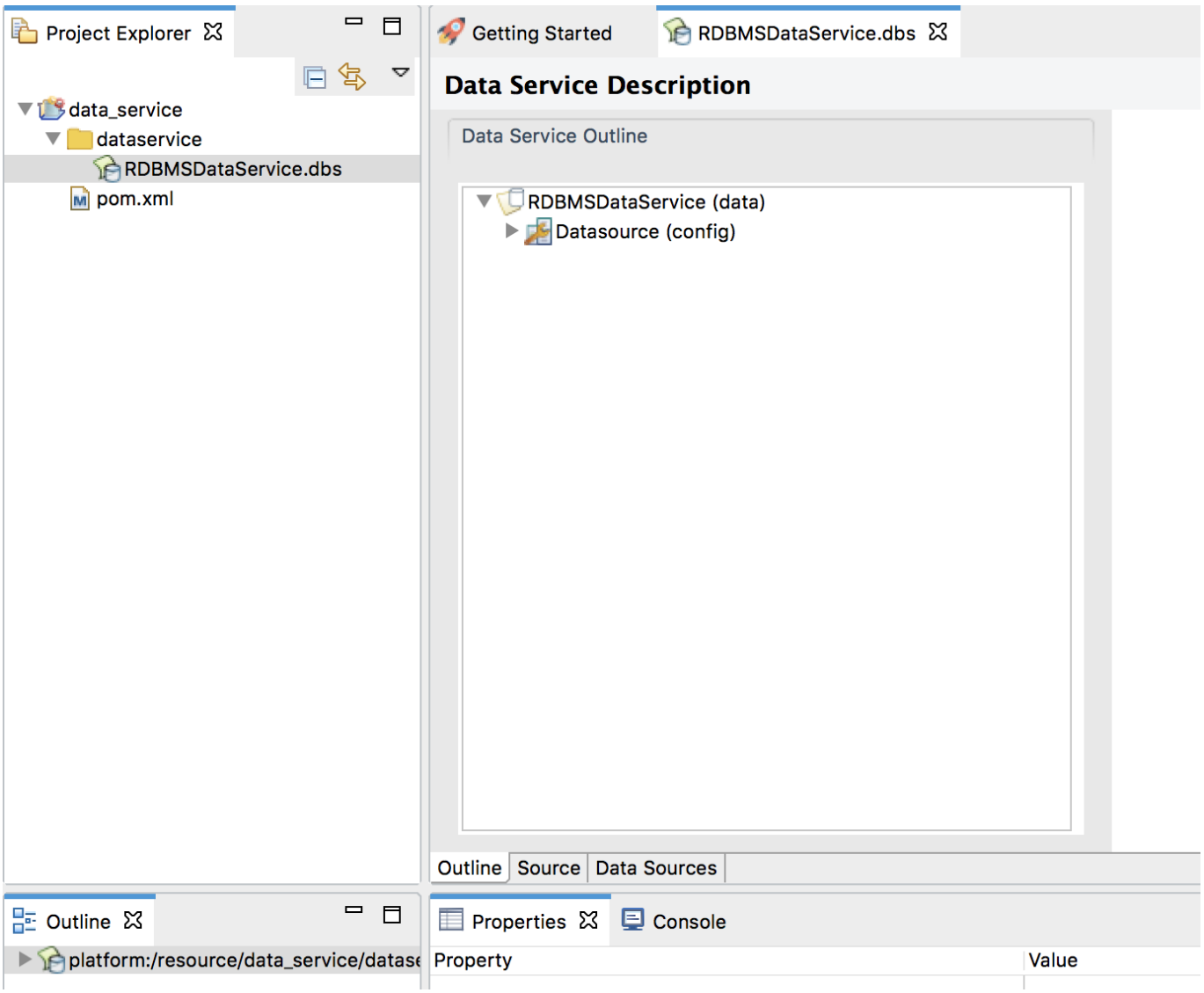

Рис. 4.6 – Вікно конфігурації Dataservice

Файл сервісу даних (.dbs) тепер буде створений у нашому Dataservice проекті.

4.3 Конфігурація ESB Config

4.3.1 Створення REST API

Для створення API клікаємо правою кнопкою миші по створеному проекту у навігаторі та переходимо до → REST API як показано на рисунку 4.7.

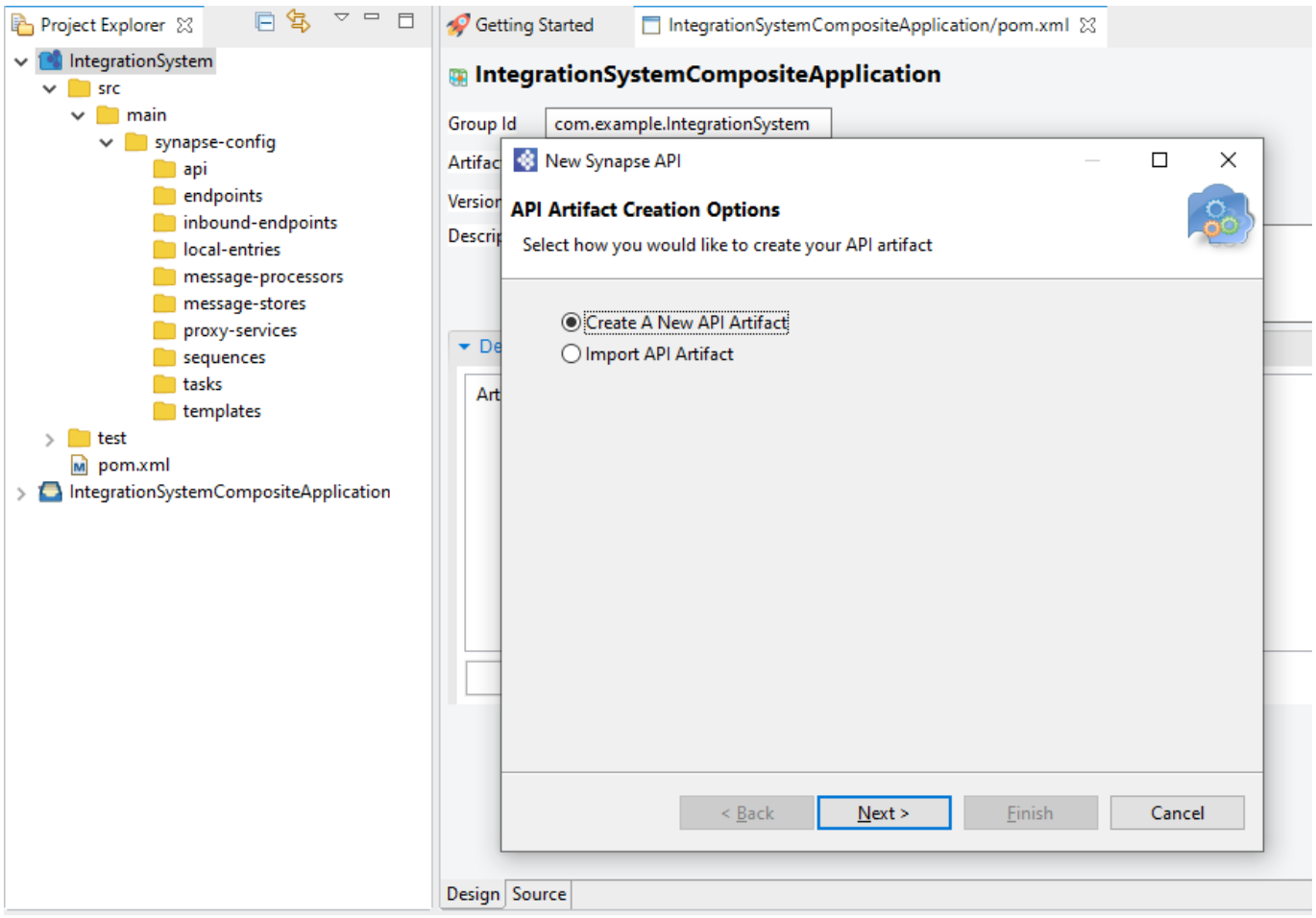

Рис. 4.7 – Створення REST API в Integration Studio

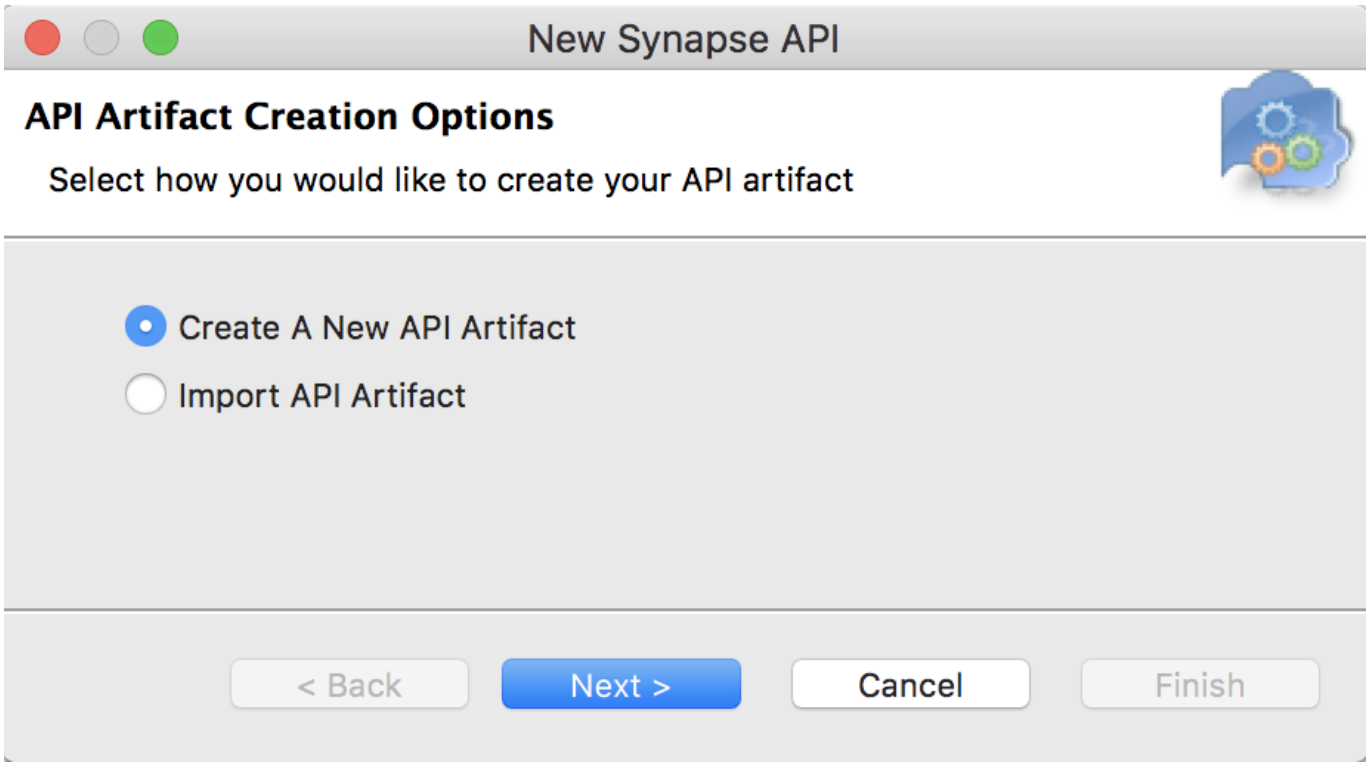

Рис. 4.8 – Створення REST API в Integration Studio

Обираємо Create A New API Artifact і переходимо далі, як показано на рисунку 4.8.

Далі необхідно заповнити обов’язкові поля для створення API (Name та Context), продемонстровано на рисунку 4.9.

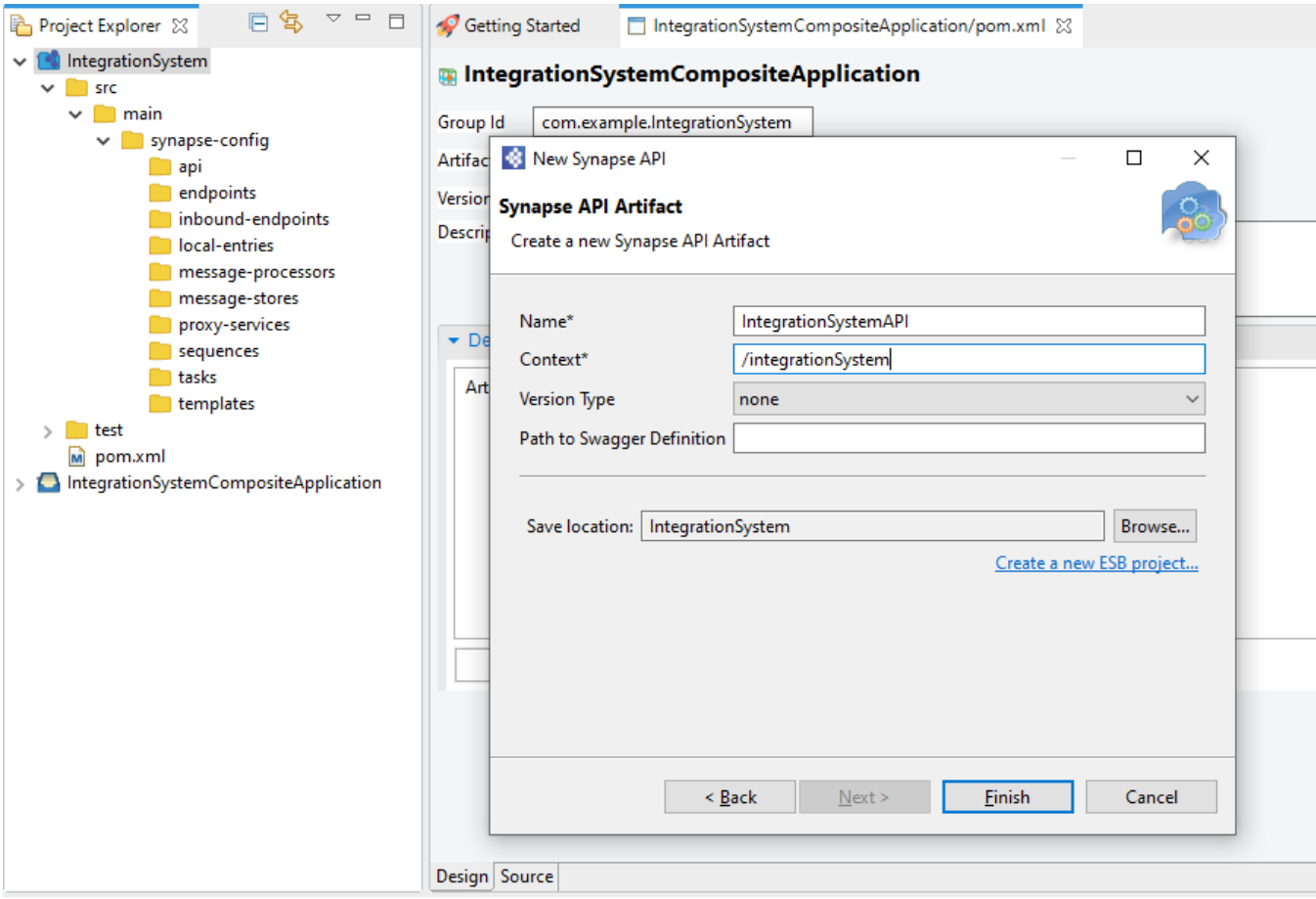

Рис. 4.9 – Створення REST API в Integration Studio

Натискаємо Finish. API REST створюється всередині папки src/main/synapse-config/api в створеному проекті.

На рисунках 4.10 – 4.11 показано новостворене API в візуальному та кодовому виді.

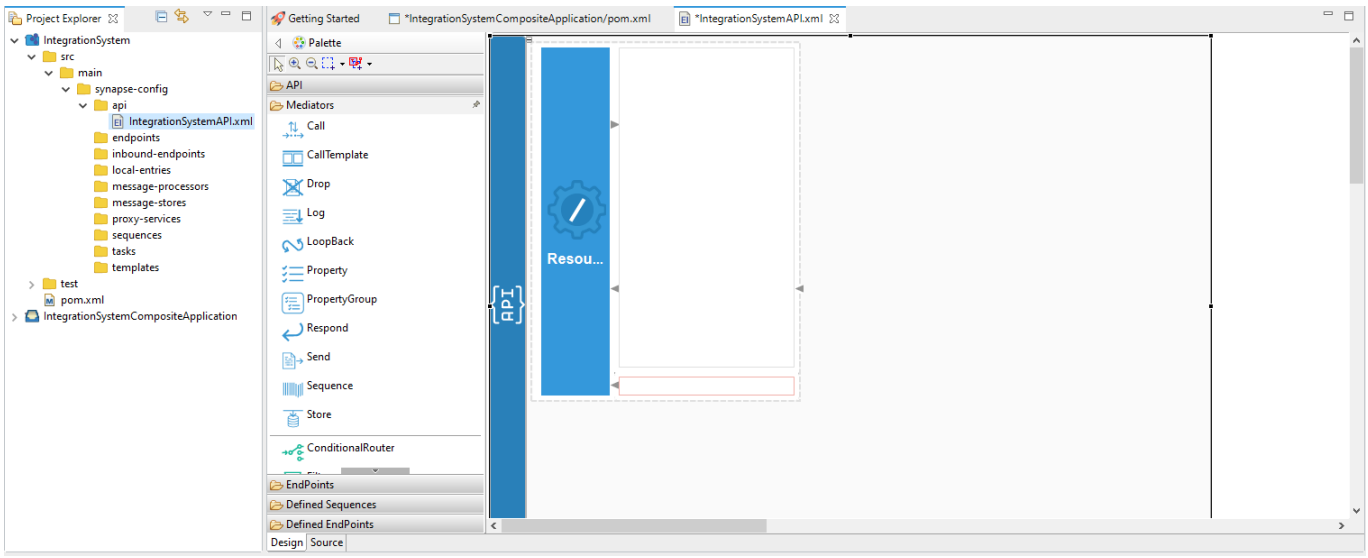

Рис. 4.10 – REST API в візуальному редакторі

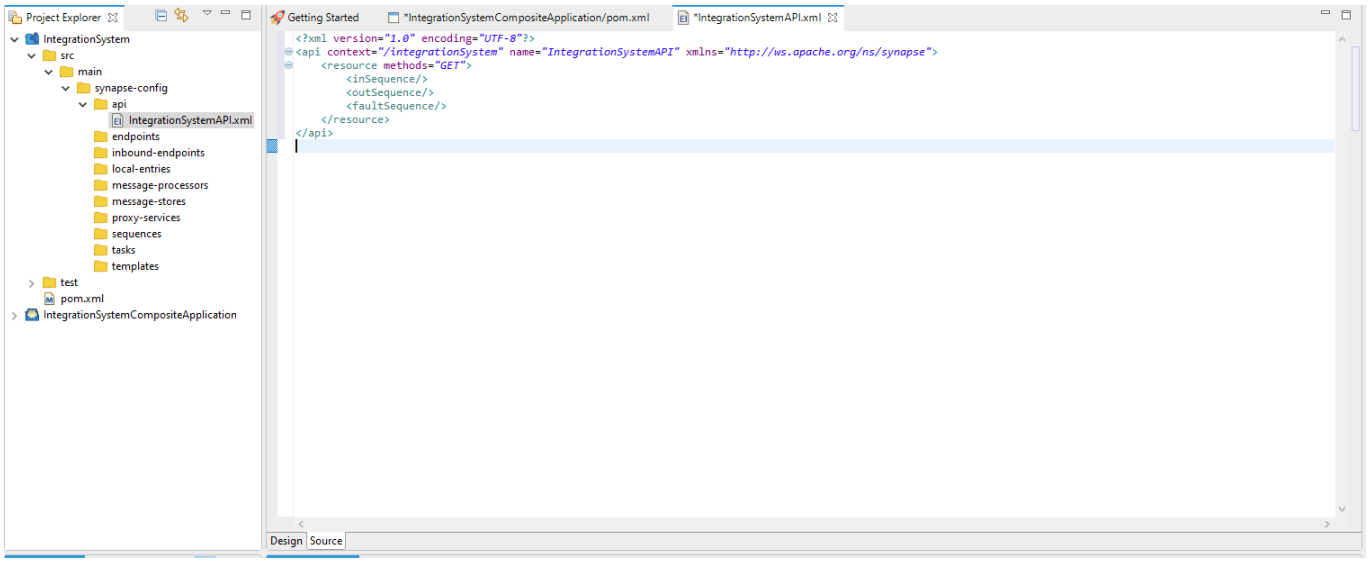

Рис. 4.11 – REST API в кодовому редакторі

Вся подальша логіка обробки запиту до системи методом get по адресу [Micro Integrator адрес:8290]/integrationSystem буде описана в середині блоку ресурсу.

4.3.2 Створення запитів для різних джерел інформації

Для отримання інформації з різних джерел данних будуть використовуватись різні медіатори та артефакти.

Для отримання інформації з WEB-сервісу чи якогось Saas по http необхідно створити Endpoint, показано на рисунках 4.12 – 4.13

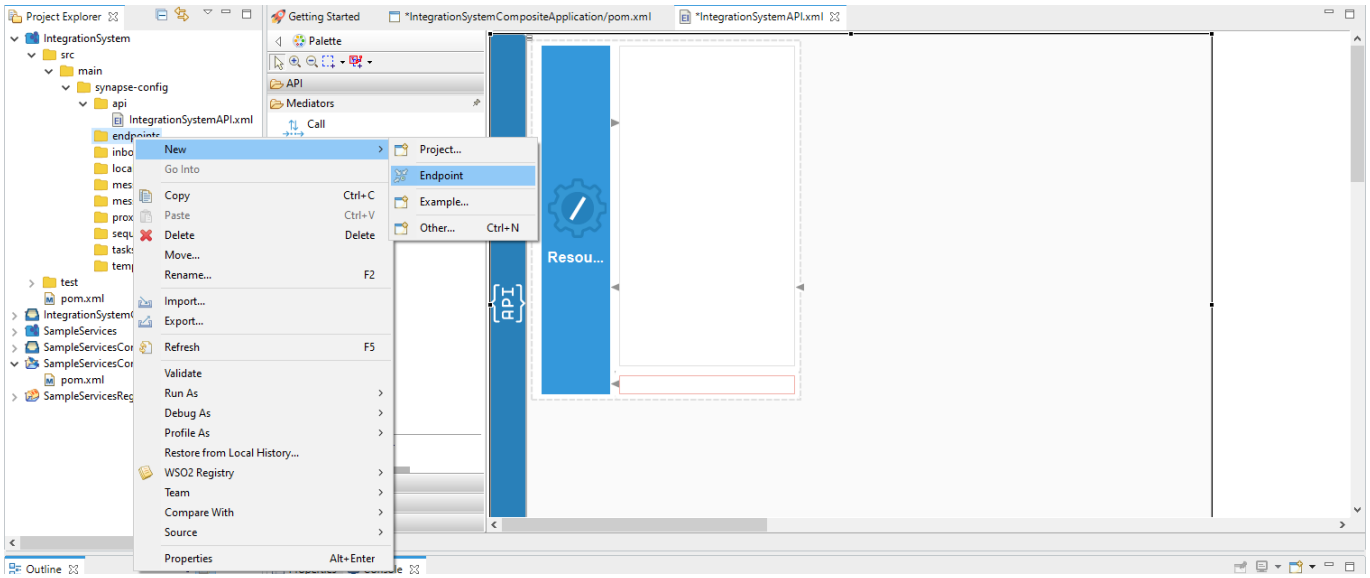

Рис. 4.12 – Створення Endpoint

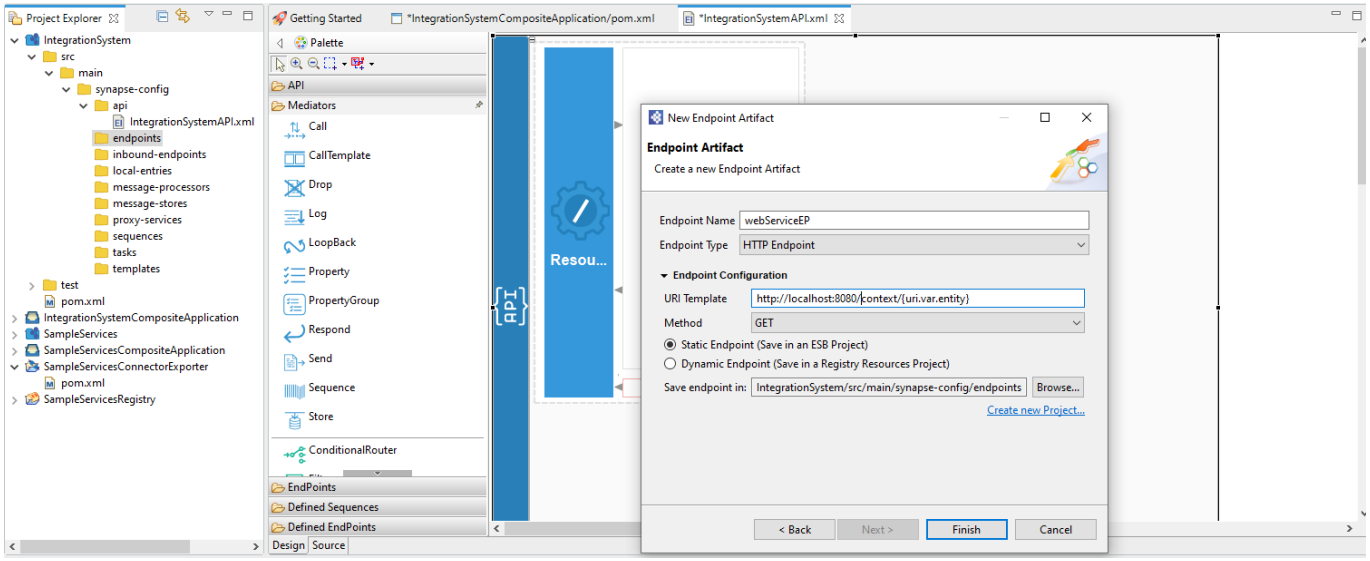

Рис. 4.13 – Створення Endpoint

Для запитів до баз даних буде створено Dataservice, описано в підрозділі 4.4

4.3.3 Паралельне виконання запитів до гетерогенних джерел даних

Для того щоб виконати запити до бази даних чи до web-сервісу було створено dataservice та endpoint відповідно, залишається сконфігурувати медіатори які будуть виконувати їх.

Для створення запиту є декілька медіаторів, таких як:

- Call mediator;
- Callout mediator;
- Send mediator;

Для виконання паралельних запитів необхідно використати медіатор clone, для створення копій вхідних повідомлень і створити n target блоків, де n – кількість джерел даних.

4.3.4 Агрегація результатів запитів

Для агрегації результатів буде використано aggregate медіатор, який буде очікувати отримання відповідей від всіх запитів за допомогою встановлення таких параметрів для властивості completeCondition:

- messageCount min="-1";
- messageCount max="-1".

4.3.5 Уніфікація результатів запитів

Кожне з джерел віддасть відповідь в своєму форматі, для приведення всіх відповідей до одного формату буде використовуватись медіатор payloadFactory, хоча для такої задачі є і інші варінти, такі як:

- використання медіатору DataMapper;
- використання медіатору Smooks(для роботи з CSV);
- використання медіатору XSLT;
- використання медіатору FastXSLT.

Приклад маппінгу показано на рисунку 4.14.

```
<payloadFactory media-type="xml">
    <format>
        <m:Field xmlns:m="http://services.samples/xsd">
            <m:Field1>$1</m:Field1>
            <m:Field2>$2</m:Field2>
        </m:Field>
    </format>
        <args>
            <arg xmlns:m0="http://services.samples/xsd" expression="//m0:someField1"/>
            <arg xmlns:m0="http://services.samples/xsd" expression="//m0:someField2"/>
        </args>
</payloadFactory>
```

Рис. 4.14 – Приклад маппінгу

4.3.6 Реалізація відповіді на запит в форматах XML чи JSON

За замовчуванням всі відповіді формуються в форматі XML. Але за необхідністю можна трансформувати до JSON за допомогою додавання властивості messageType з параметрами як показано нижче:

<property name = "messageType" value = "application/json" scope = "axis2" type = "STRING"></property>

Але споживач API буде мати можливість прочитати відповідь в форматі XML, якщо встановить такий заголовок в http запит:

Accept = application/xml

4.4 Конфігурація Dataservice

Після того як ми створили проект датасервісу в підрозділі 4.2 необхідно правильно сконфігурувати datasource для баз даних. Для прикладу взято одну SQL БД і одну NoSQL.

4.4.1 Конфігурація datasource для MySQL

Конфіг datasource для БД MySQL показано на рисунку 4.15.

```
<config id="MySQL_ds">
    <property name="org.wso2.ws.dataservice.user">admin</property>
    <property name="org.wso2.ws.dataservice.password">admin</property>
    <property name="org.wso2.ws.dataservice.protocol">jdbc:mysql://localhost:3306/someDB</property>
    <property name="org.wso2.ws.dataservice.driver">com.mysql.jdbc.Driver</property>
</config>
```

Рис. 4.15 – Конфіг для БД MySQL

Крім конфігу потрібно описати ще такі два елемента як operation і query

Query – це запити до самої БД, які містять в собі текст запиту з вхідними параметрами, типи вхідних і вихідних даних і маппінг результатів якщр необхідно.

Operation – стосується операції веб-сервісу, визначеної запитом. Операція визначається як виклик запиту, що вказує, як параметри запиту обчислюються або виводяться.

Приклад query для пошуку даних в таблиці things з використанням datasource MySQL_ds показано на рисунку 4.16.

```
<query id="select_all_query" useConfig="MySQL_ds">
    <sql>SELECT field1, field2 FROM THINGS</sql>
    <result element="Documents" rowName="Document">
        <element column="FIELD1" name="FIELD1" xsdType="xs:string"/>
        <element column="FIELD2" name="FIELD2" xsdType="xs:string"/>
    </result>
</query>
```

Рис. 4.16 – Query для БД MySQL

Приклад operation, що буде ініціювати запит select_all_query показано на рисунку 4.17.

```
<operation name="select_all_operation">
   <call-query href="select_all_query"/>
</operation>
```

Рис. 4.17 – Operation для БД MySQL

4.4.2 Конфігурація datasource для MongoDB

Конфіг datasource для БД монго показано на рисунку 4.18.

```
<config id="mongo_ds">
    <property name="mongoDB_servers">localhost</property>
    <property name="mongoDB_database">somedb2</property>
</config>
```

Рис. 4.18 – Конфіг для БД mongodb

Приклад query для пошуку даних в колекції things з використанням datasource mongo_ds показано на рисунку 4.19.

```
<query id="mongo_find" useConfig="mongo_ds">
    <expression>things.find()</expression>
    <result element="Documents" rowName="Document">
       <element column="document" name="Data" xsdType="string"/>
    </result>
</query>
```

Рис. 4.19 – Query для БД mongodb

Приклад operation, що буде ініціювати запит mongo_find показано на рисунку 4.20.

```
<operation name="mongo_find">
   <call-query href="mongo_find"> </call-query>
</operation>
```

Рис. 4.20 – Operation для БД mongodb

4.5 Послідності проекту

Послідовність(Sequence) може бути безпосередньо викликана з будь-якої точки, і можна викликати послідовність з іншої послідовності. Також можна налаштувати весь потік повідомлень та логіку обробки повідомлень в одній послідовності, але найкраще розробити сценарії інтеграції таким чином, щоб послідовності стосувалися різних частин великого сценарію інтеграції. Це розкладе складний випадок використання повідомлень на прості блоки обробки повідомлень та сприятиме повторному використанню та усуненню проблем.

Є три послідовності за замовчуванням:

- inSequence;
- outSequence;
- faultSequence.

Вхідна послідновність(inSequence) – початкова точка всіх вхідних повідомлень, які попали в інтеграційний потік. Від використання різних медіаторів залежить чи буде передано керування в вихідну послідовність(outSequence).

Часто інтеграцію реалізують без використання вихідної послідновності, тобто весь процес обробки поміщають в вхідну послідовність.

Також існує faultSequence – послідовість опрацювання помилок. Всі внутрішні помилки ESB, які були визвані під час опрацювання повідомлення передають подальше опрацювання повідомлення до faultSequence.

4.5.1 Вхідна послідовність

Візуально вхідну послідовність можна представити в вигляді блок-схеми, показаної на рисунку 4.21 та в Додатку Д.

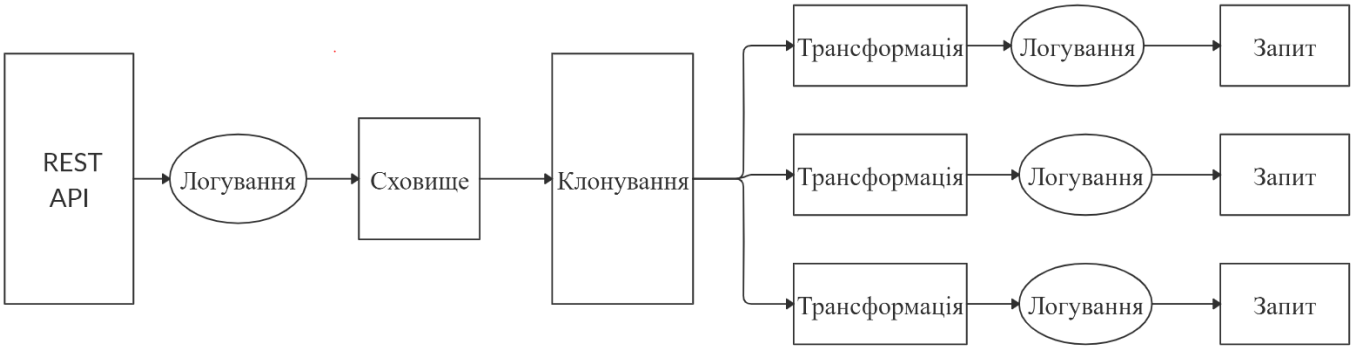

Рис. 4.21 – Блок-схема вхідної послідовності

Кінцева точка REST API – сконфігурований медіатор Resource, в якому вказаний url-адрес та метод запиту.

Логування – сконфігурований медіатор Log, налаштований на запис всього повідомлення за допомогою властивості level значення якої є full.

Сховище – сконфігурований медіатор Store, в якому вказана черга повідомлень на топік.

Клонування – сконфігурований медіатор Clone, налаштований на подальше паралельне опрацювання.

Трансформація – сконфігурований медіатор payloadFactory, в якому відбувається трансформація вхідного повідомлення до необхідного формату для подальшому запиті.

Запит – сконфігурований медіатор Send, який виконує запит по необхідному адресу, та передає подальше виконання до вихідної послідовності.

4.5.2 Вихідна послідовність

Візуально вхідну послідовність можна представити в вигляді блок-схеми, показаної на рисунку 4.22 та в Додадтку Е.

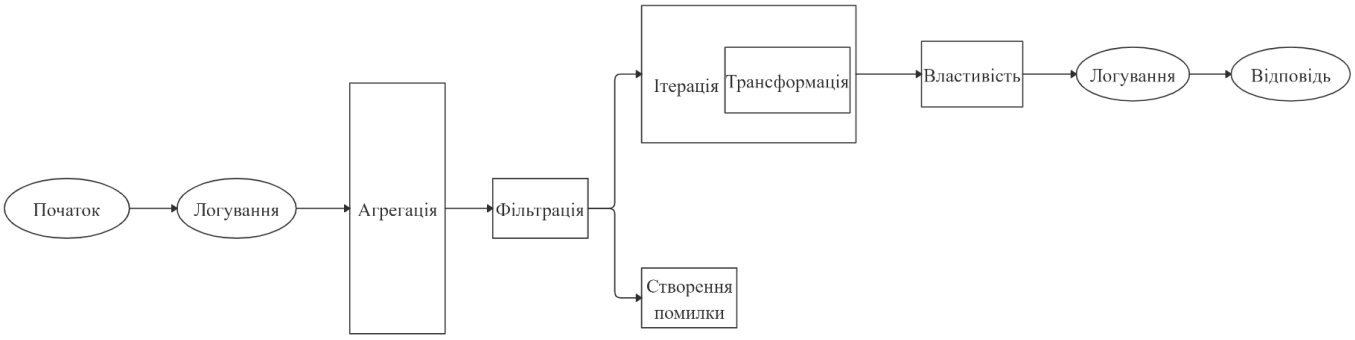

Рис. 4.22 – Блок-схема вихідної послідовності

Медіатор Send з вхідної послідовності налаштований на передачу подальшу опрацювання повідомлення до вихідної послідовності.

Агрегація – сконфігурований медіатор Aggregate, який налаштований на очікування всіх запитів, що були створені в вхідній послідовності.

Фільтрація – сконфігурований медіатор filter, налаштований на перевірку кількості отриманих результатів. Якщо повідомлення успішно пройшло через фільтр то повідомлення передається до блоку ітерації, а якщо фільтр не пройдено – передача до блоку «створення помилки».

Створення помилки – сконфігурований медіатор makeFault, який передає подальшу обробку повідомлення до послідовності faultSequence.

Ітерація – сконфігурований медіатор iterate, який циклічно виконує трансформацію для кожного повідомлення, отриманих медіатором aggregate і які пройшли фільтр.

Властивість – сконфігурований медіатор property, в якому ми добавляємо статус-код для подальшої http відповіді.

Відповідь – сконфігурований медіатор respond, який віддасть відповідь клієнту разом зі всіма даними та статус-кодом про усішність виконання операції.

4.5.3 Послідовність опрацювання помилок

Візуально послідовність обробки помилок можна представити в вигляді блок-схеми, показаної на рисунку 4.23 та в Додатку Ж.

Рис. 4.23 – Блок-схема послідовності обробки помилок

Медіатор makeFault з вихідної послідовності ініціює подальшу обробку повідомлення до послідовності обробки помилок.

Логування – сконфігурований медіатор Log, налаштований на запис всього повідомлення.

Властивість – сконфігурований медіатор property, в якому ми добавляємо статус-код для подальшої http відповіді.

Відповідь – сконфігурований медіатор send, який віддасть відповідь клієнту разом зі всіма даними та статус-кодом про усішність виконання операції.

4.6 Створення артефактів для роботи зі сховищем повідомлень

Для роботи зі сховищами повідомлень необхідно створити два артефакти:

- Message Store;
- Message Processor.

Message Store використовується для тимчасового зберігання повідомлень, перш ніж вони будуть доставлені до місця призначення процесором повідомлень. Цей підхід корисний для обслуговування трафіку для бек-енд-сервісів, які можуть приймати повідомлення лише з заданою швидкістю, тоді як вхідний трафік до ESB надходить з різною швидкістю. Щоб зберігати вхідний трафік у магазині повідомлень, використовується медіатор Store, а потім робота переходить до процесора обробки повідомлень для доставки повідомлень до бек-енд-сервісу із фіксованою швидкістю.

Кілька процесорів повідомлень можуть використовувати одне сховище повідомлень. Наприклад, у кластерному середовищі кожен із вузлів мав би екземпляр

одного і того ж процесора повідомлень, кожен з яких би з'єднувався з тим самим сховищем повідомлень і рівномірно споживав з нього повідомлення. Message Store діє як менеджер цих споживачів та їх з'єднань і забезпечує обробку повідомлень лише одним процесором повідомлень, запобігаючи дублюванню повідомлень.

Використання процесорів повідомлень та сховищ повідомлень дозволяє реалізувати різні схеми обміну повідомленнями та інтеграції.

Блок-схема гарантованої доставки повідомлень за допомогою використання черги повідомлень наведену на рисунку 4.24 та в Додатку К.

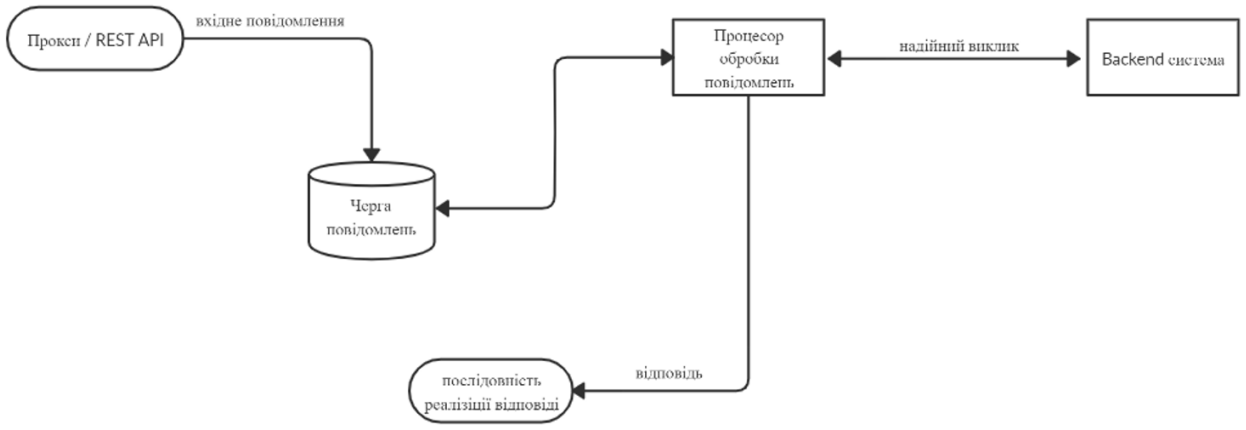

Рис. 4.24 – Блок-схема гарантованої доставки повідомлень

4.6.1 Створення артефакту Message Store

Для створення Message Store клікаємо правою кнопкою миші по створеному проекту у навігаторі та переходимо до → Message Store після чого появляється вікно створення сховища, яке показано на рисунку 4.25.

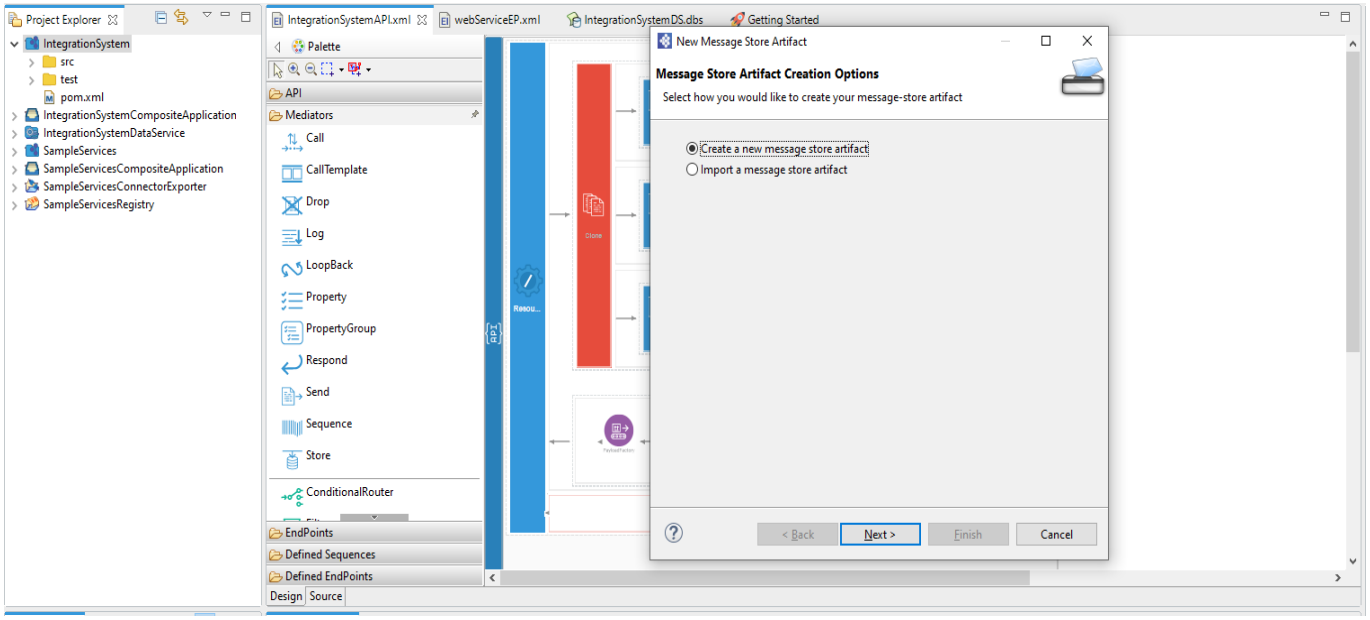

Рис. 4.25 – Створення сховища повідомлень

Після чого натискаємо Next і попадаємо до конфігурації сховища, продемонстровано на рисунку 4.26, де потрібно вказати такі обов’язкові поля як ім’я для створюваного сховища, його тип. Далі залежно від типу сховища необхідно буде заповнити різні поля.

Без додадткових налаштуваня за замовчуванням доступні такі типи сховищ даних:

– JMS Message Store;
– In Memory Message Store;
– Custom Message Store;
– RabbitMQ Message Store;
– JDBC Message Store;
– WSO2 MB Message Store;
– Resequence Message Store.

На рисунку 4.25 продемонстрована конфігурація для сховища RabbitMQ.

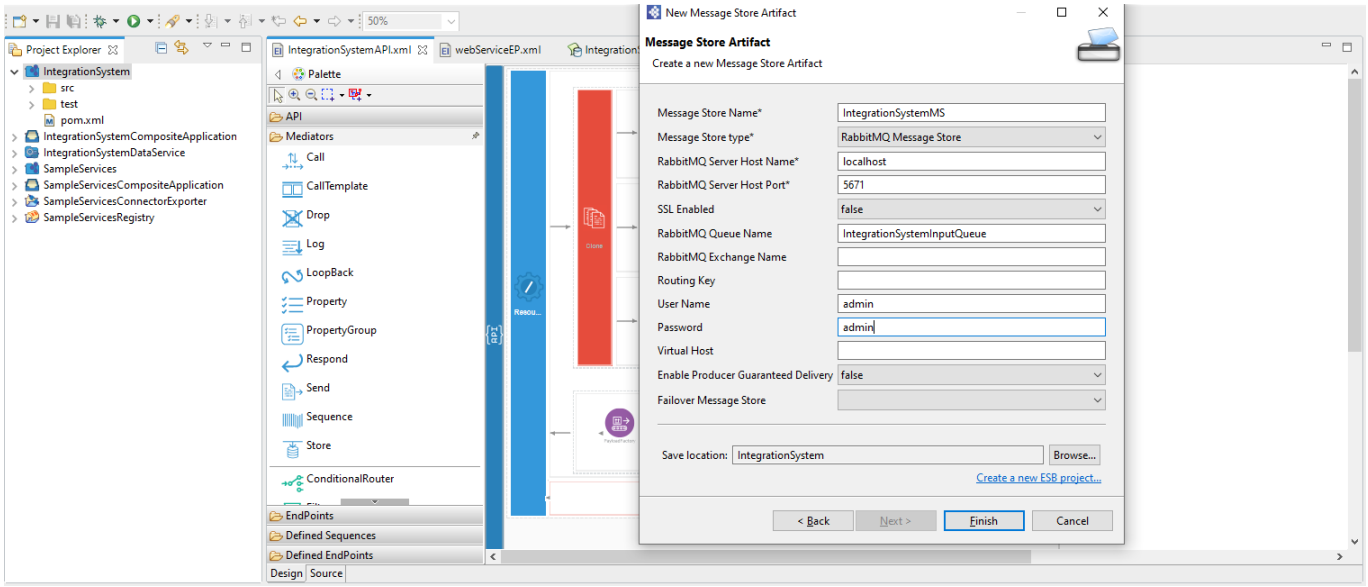

Рис. 4.26 – Конфігурація сховища RabbitMQ

Натискаємо на Finish і у нас створено сховище даних RabbitMQ яке можна побачити в директорії synapse-config/message-stores .

4.6.2 Створення артефакту Message Processor

Створюємо процесор типу Scheduled Message Forwarding Processor.

Scheduled Message Forwarding Processor – це процесор повідомлень, який споживає повідомлення зі сховища і надсилає їх до кінцевої точки. Якщо повідомлення успішно доставлено до кінцевої точки, процесор видаляє повідомлення з сховища повідомлень. У разі невдачі він повториться через визначений інтервал.

Для створення Message Processor клікаємо правою кнопкою миші по створеному проекту у навігаторі та переходимо до → Message Processor після чого появляється вікно створення процесора обробки повідомлень, яке показано на рисунку 4.27.

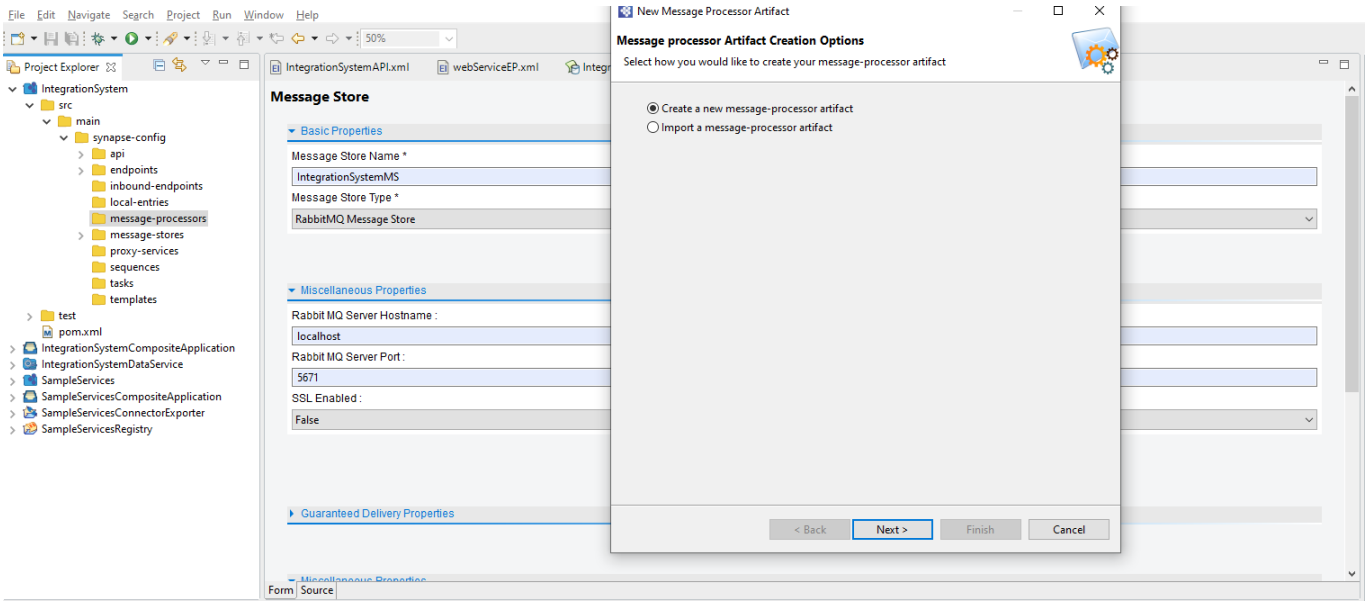

Рис. 4.27 – Створення процесора обробки повідомлень

Після чого натискаємо Next і попадаємо до конфігурації процесора. Сконфігуровані такі поля як:

– Message processor type;
– Message processore name;
– Message Store;
– Processor State;
– Forwarding interval;
– Retry interval;
– Maximum redelievery attempts;
– Maximum store connection attempts;
– Store connection attempt interval;
– Drop message after maximum attempts;
– Task count;
– Endpoint name;
– Fail message store.

Необхідно сконфігурувати Scheduled Message Forwarding Processor для сховища повідомлень IntegrationSystemMS, яке було створено раніше так як показано на рисунку 4.28.

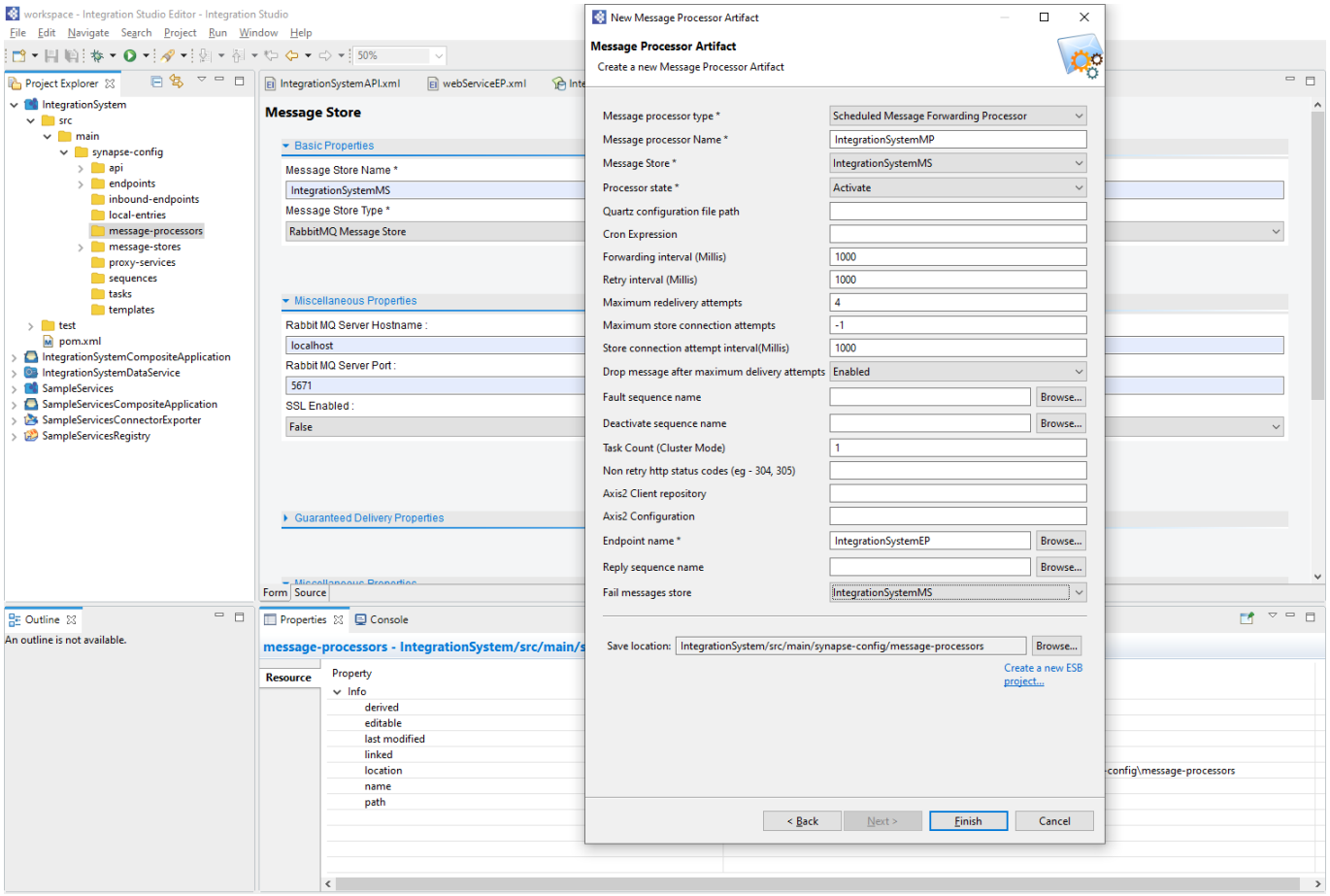

Рис. 4.28 – Сконфігурований процесор обробки повідомлень типу Scheduled Message Forwarding Processor

Як і в сховища повідомлень, процесор має декілька типів:

- Scheduled Message Forwarding Processor;
- Message Sampling Processor;
- Custom Message Processor;
- Scheduled Failover Message Forwarding Processor.

4.7 Експорт всіх артефактів та конфігурацій

Для того щоб запустити проект спочатку потрібно запакувати всі артефакти в CAR архів. Для цього потрібно вибрати composite application проект в провіднику

проектів, клацнути правою кнопкою миші та натиснути «export composite application project», що продемонстровано на рисунку 4.29.

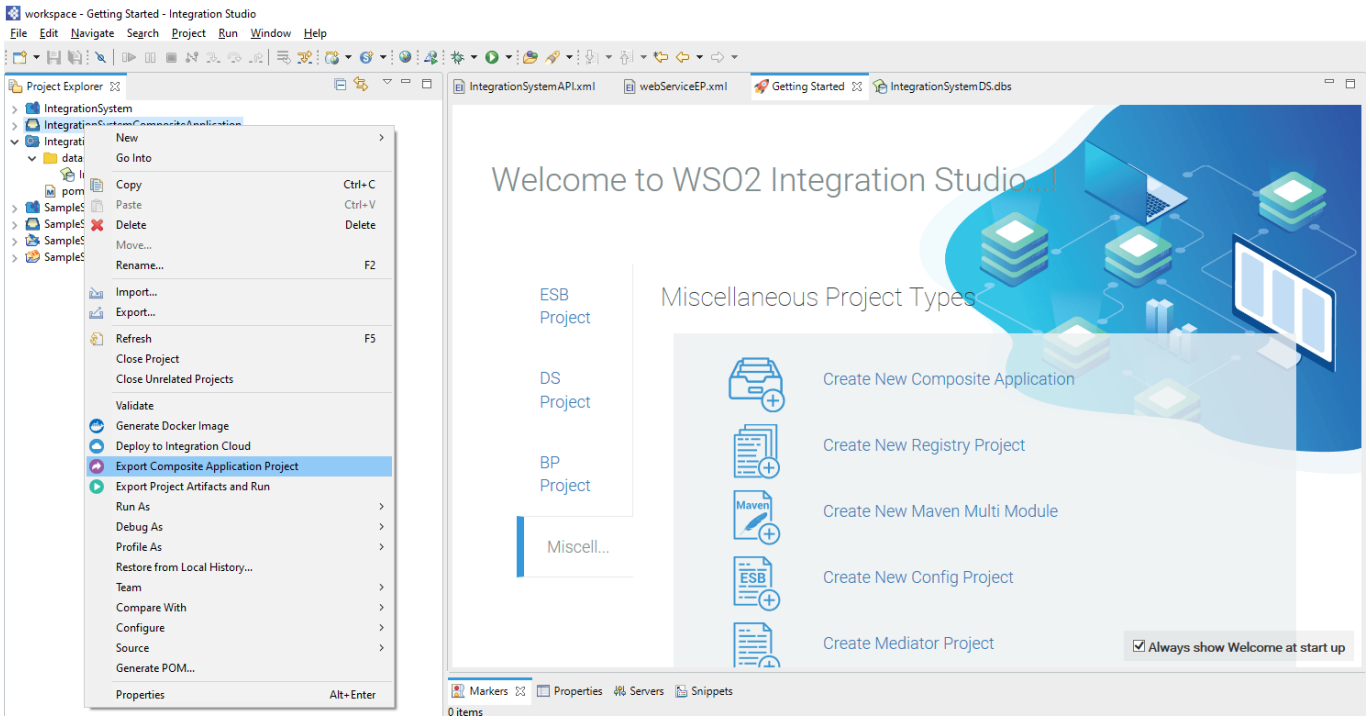

Рис. 4.29 – Експорт проекту

У діалоговому вікні, що відкриється, даємо назву для CAR архіву, місце збереження та натискаємо кнопку Next, показано на рисунку 4.30.

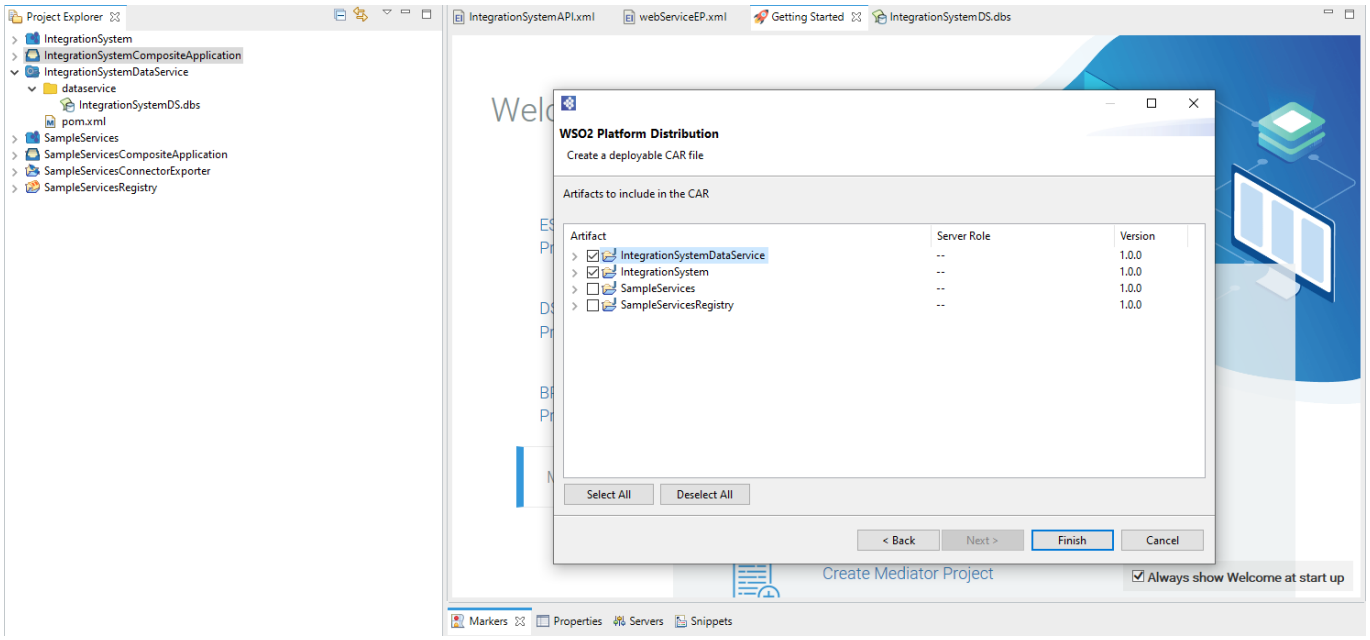

Рис. 4.30 – Вибір артефактів для пакування в архів

Клікаємо Finish для створення CAR архіву, після чого його можна імпортувати в Micro Integrator.

4.8 Розгортання системи в Micro Integrator

Після того як CAR архів було успішно експортовано є 2 варіанти розгортання інтеграційної системи на сервер Micro Integrator.

Перший варіант – просте копіювання CAR-архіву в директорію «MI_HOME/repository/deployment/server/carbonapps/».

Другий – за допомогою Integration Studio, де необхідно натиснути правою клавішою миші на «composite application project» і обрати «Export Project Artifacts and Run», показано на рисунку 4.31.

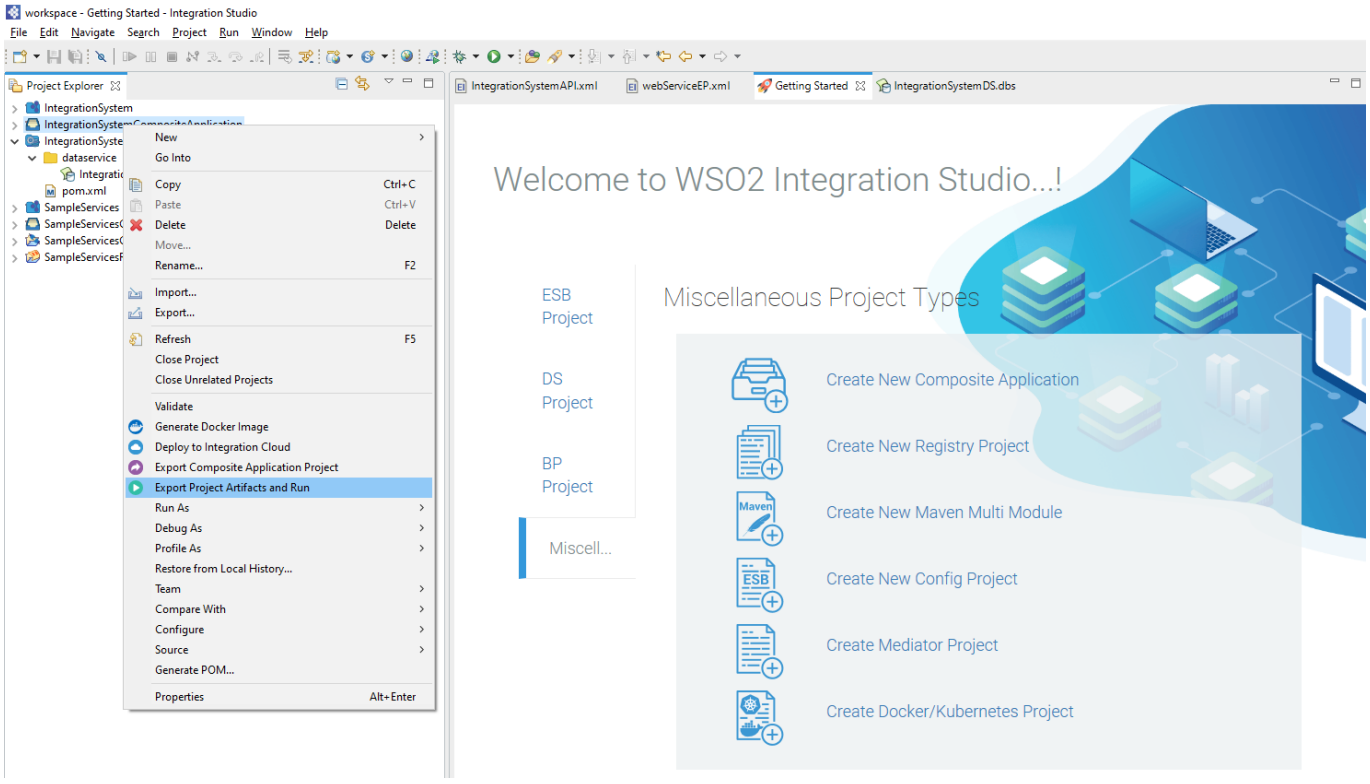

Рис. 4.31 – Розгортання CAR-архіву

Після чого необхідно вибрати необхідні артефакти для пакування і запуску та натиснути Finish.

Після цього кроку автоматично починається деплой архіву на сервер Micro Integrator, процес якого можна спостерігати в консолі Integration Studio. Показано на рисунку 4.32.

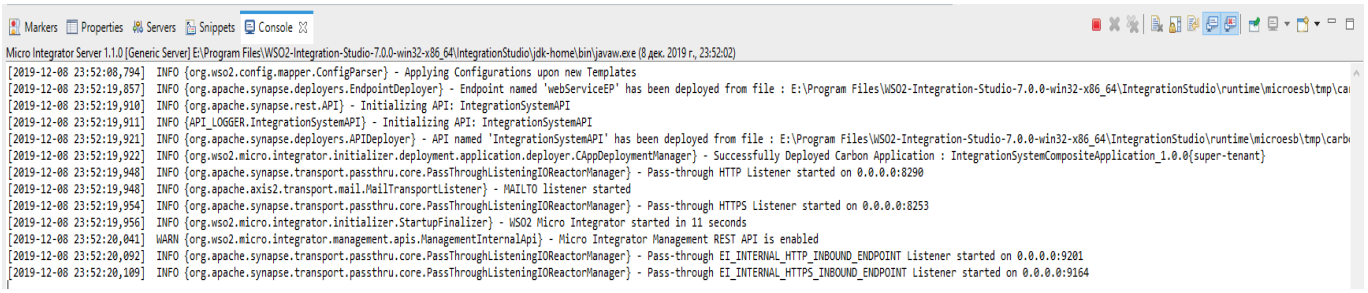

Рис. 4.32 – Процес розгортання CAR-архіву

Процес розгортання всіх необхідних слухачів та артефактів пройшов успішно.

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

5.1 Вступ

Напрацювання, викладені у даній магістерській дисертації, можуть бути практично застосовані і мають комерційну цінність для організацій з різними джерелами інформації. Дана система може бути корисною в першу чергу в наукових, освітніх, юридичних та комерційних організаціях, що орієнтовані на міжнародну співпрацю. Такі системи на даний момент відсутні на ринку.

5.2 Опис ідеї стартап-проекту

Ідея стартап-проекту описана у таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система інтеграції гетерогенних джерел даних	1. Створення системи з уніфікованим форматом даних	Створення єдиної системи для отримання даних з різних джерел

Продовження таблиці 5.1

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	2. Підготовка спеціалістів з використання системи інтеграції	1. Вивчення використання системи інтеграції. 2. Впевненість у професійності підготовлених спеціалістів.
	3. Інтегрування системи інтеграції в підприємство.	Використання системи для управління уніфікованими даними з різнорідних джерел інформації. Пониження використання ресурсів організації і децентралізація процесів. Можливість переносу систему на хмарні ресурси.

Аналіз сильних, слабких та нейтральних сторін проекту наведено у таблиці 5.2. Резюмуючи коротко, перевагами системи є великий набір готових конекторів для роботи з багатьма інформаційними системами, кросплатформенність (не вимагає серверів з пропрієтарним програмним забезпеченням) та можливість переносу системи на хмарні ресурси.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

		(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	IBM	Oracle			
1	Вартість експлуатації	500 грн за місяць	1666 доларів США/місяць ліцензія	1200 доларів США/місяць ліцензія	-	-	+
2	Кросплатформенність	+	+	+	-	+	-
3	Наявність технічної підтримки на українській, російській та англійській мові	+	+	–	-	-	+
4	Наявність можливості роботи з різних пристроїв	+	+	+	-	+	-
5	Форм-фактори системи	+	+	-	-	+	-

Продовження таблиці 5.2

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	IBM	Oracle			
6	Стійкість до відмов	Середня	Висока	Середня	+	-	-
7	Можливість балансування навантаження	+	-	-	-	-	+
8	Зручність інтеграції з іншими системами	Висока	Висока	Висока	-	+	-
9	Робота з великою кількістю сервісів	+	+	+	-	+	-
10	Складна структура інфраструктури	+	+	+	-	+	-
11	Зручність у користуванні	Висока	Висока	Висока	-	+	-
12	Зручність у адмініструванні	Низька	Низька	Низька	-	+	-

Продовження таблиці 5.2

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	IBM	Oracle			
13	Швидкість обчислень	Висока	Висока	Висока	-	+	-
14	Налаштування прав доступу	+	+	+	-	+	-

5.3 Технологічний аудит проекту

Аналіз складових технологічного процесу наведено у таблиці 5.3.

Таблиця 5.3

№ п\п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1		WSO2 EI	Наявні	Доступні
2		IBM App Connect	Наявні	Доступні
3		Oracle ESB	Наявні	Доступні

Продовження таблиці 5.3

№ п\п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
4		Mule ESB	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: WSO2				
5		WSO2 EI	Наявні	Доступні
6		WSO2 MI	Наявні	Доступні
7		WSO2 API-M	Наявні	Доступні
8		WSO2 DSS	Наявні	Доступні
9		WSO2 DAS	Наявні	Доступні

В цілому, використовуються майже всі технології продукту WSO2, що працюють з даними.

5.4 Аналіз ринкових можливостей запуску проекту

В цілому, ринок готовий до сприйняття продукту, і продукт має свою нішу на ринку. Це підтверджується попереднім оглядом ринку, наведеному у таблиці 5.4.

Умовною одиницею продажу є ліцензія на 1 місяць на 1 користувача.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, одиниць	4
2	Загальний обсяг продаж, грн/ум.од	500 грн/ум.од
3	Динаміка ринку (якісна оцінка)	Зростає

Продовження таблиці 5.4

№ п/п	Показники стану ринку (найменування)	Характеристика
4	Наявність обмежень для входу (вказати характер обмежень)	Недискримінацій ні якісні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні на початковому етапі
6	Середня норма рентабельності в галузі (або по ринку), %	80%

Потенційні характеристики клієнтів стартап-проекту описані у таблиці 5.5. Це можуть бути як приватні, так і державні установи.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Створення системи з уніфікованим форматом даних		Створенная системи, орієнтовану на потреби цільвих груп клієнтів, для надання швидкого та зручного доступу до даних	Визначення умов яким має відповідати інтеграційна система

Продовження таблиці 5.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
2	Підготовка спеціалістів з використання системи інтеграції		Знаходження персоналу для вивчення інтеграційної системи	1. Вивчення використання системи інтеграції. 2. Впевненість у професійності підготовлених спеціалістів.
3	. Інтегрування системи інтеграції в підприємство		Допомога у створенні інтеграційної системи	Використання системи для управління уніфікованими даними з різнорідних джерел інформації. Пониження використання ресурсів організації і децентралізація процесів. Можливість переносу систему на хмарні ресурси.

Проте, український ринок і ринок інформаційних послуг в цілому є нестабільним і необхідним є врахування низки загрозливих факторів. Фактори ризику і загрози на ринку описані у таблиці 5.6.

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Непередбачувані зміни законодавства	Зміна вимог до організації з боку органів державної влади (наприклад, рівень оподаткування)	Організація юридичної особи в країні зі сприятливим інвестиційним кліматом
2	Відсутність інтернет мережі	Без відсутності мережі, підприємство не зможе обезпечити роботу публічного API системи	Підписання надійного контракту з інтеренет провайдером, для стабільної роботи мережі.
3	Недостатня швидкість інтернет з’єднання	Через недостатньо швидке інтернет з’єднання, пересилання і отримання данних може відбуватися з затримкою.	Підписання надійного контракту з інтеренет провайдером, для надання високої швидкості інтернет з’єднання.

Продовження таблиці 5.6

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
5	Недовіра користувачів	Користувачі не довіряють новому гравцю на ринку	Впровадження на безоплатній основі в одній з організацій

Проте, наразі ринок сформував також низку можливостей, реалізація яких дозволить отримати прибуток і завоювати ринок.

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Отримання необхідних інвестицій	Сформований початковий капітал, необхідний для реалізації мінімально життєздатного продукту	Розробка мінімально життєздатного продукту
2	Розвиток ринку інтеграційних інфраструктур	Установи що активно впроваджують інформаційну інфраструктуру	Надання послуг державним організаціям, участь у тендерах
3	Зростання промисловість	Незважаючи на складнощі, ВВП країни зростає, на	Робота з новими підприємствами на ринку, реорганізація

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
		ринку з’являються нові підприємства	процесу роботу з даними у існуючих
5	Успішна маркетингова та інформаційна політика	Інформаційна кампанія з прикладами успішного впровадження системи	Збільшення кількості споживачів при фактично сталій вартості розробки та підтримки системи

Далі було розглянуто конкуренцію на ринку. Результати ступеневого аналізу подані у таблиці 5.8.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Олігополія	Незначна кількість конкурентів Велика ринкова сила Схожість використовуваних технологій	Інформування ринку щодо появи нової веб-служби
Галузевий	Загроза появи нових конкурентів Ринкова влада споживачів	Інформування ринку щодо якості використовуваної новаторської технології Пропозиція гнучких цін

Продовження таблиці 5.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
	Висока потреба у товарі	
Внутрішньогалузева	Діяльність в одній галузі економіки Надання сервісів одного типу	Зменшення вартості сервісу Примноження каналів розподілу
Товарно-видова	Надання різних сервісів одного виду	Маркетингова політика
Цінова	Використання цін для покращення економічних умов збуту	Зменшення вартості сервісу Використання нових каналів розподілу
Марочна	Пропозиція схожого сервісу Спільна цільова аудиторія	Інформування ринку щодо якості використовуваної новаторської технології Примноження каналів розподілу

Більш детальний аналіз конкуренції на ринку було виконано за моделлю 5 сил М. Портера, результати наведені у таблиці 5.9.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складовий аналізу	IBM App Connect	Капіталовкладення, існуючі напрацювання в кожній з поєднуваних компонент	Відсутні	- Змінні витрати: Виробничі непрямі дегресивні - Системи інформації: реклама та директ-маркетинг, - Рівень чутливості до цін: орієнтовані на цінність продукту - Продуктова диференціація: якість, спосіб отримання сервісу, швидкість обслуговування Методи контролю якості: ручне та автоматичне	Копіювання функціоналу, Монополізація дистриб'юторів, Демпінг

Продовження таблиці 5.9

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
				тестування, профілювання, прототипування, рев'ю коду, аудит	
Висновки	CR4 = 85% Індекс Херфіндаля-Хіршмана (HHI) = 3450 Значення показників вказує на високу концентрацію (монополізацію) даного ринку	Можливості входу на ринок забезпечить мінімізація цін, швидкість та простота надавання послуги споживачам, швидке реагування на вимоги користувачів. В результаті аналізу проектів на народно-громадських інтернет-платформах потенційних конкурентів	Відсутні	Клієнти диктують умови гнучкості цінової політики, високої і довгострокової якості послуг та підтримку всіх видів джерел інформації з якими працюють	Пропонування вигідних умов безпосереднім користувачам, виділення ліній підтримки

Продовження таблиці 5.9

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
		знайдено не було			

Незважаючи на високу конкуренцію на ринку і наявних на ньому гравців, пропонована система має низку переваг, що описані у таблиці 5.10.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Унікальність сервісу	Створення системи інтеграції гетерогенних джерел даних
2	Ціна	Аналоги є невиправдано дорогими.
3	Цінова політика	Отримання прибутку здійснюється за рахунок продажу ліцензій, що є звичайною політикою у галузі
4	Додаткові послуги	Можливе розширення функціональності під вимоги конкретного користувача, в тому числі безкоштовно – якщо таке розширення збільшує цінність системи для потенційних клієнтів

За визначеними факторами конкурентоспроможності, наведеними у таблиці 5.10, було виконано порівняльний аналіз сильних та слабких сторін стартап-проекту. Результати аналізу наведені у таблиці 5.11.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Рейтинг товарів-конкурентів у порівнянні з моєю системою						
		–3	–2	–1	0	+1	+2	+3
1	Унікальність сервісу						+	
2	Ціна							+
3	Цінова політика							+
4	Додаткові послуги							+

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу на основі раніше отриманих даних.

SWOT-аналіз полягає у побудові матриці, що включає:

- сильні сторони проекту (Strength);
- слабкі сторони проекту (Weak);
- загрози (Troubles);
- можливості (Opportunities).

На основі SWOT-аналізу були розроблені альтернативи ринкової поведінки для виведення стартап-проекту на ринок та розрахований орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.

Результати проведеного SWOT-аналізу наведені у таблиці 5.12.

Таблиця 5.12 – SWOT- аналіз стартап-проекту

Сильні сторони:	Слабкі сторони:
Гнучка цінова політика Додаткові сервіси	Нестача стартових капіталовкладень Сильна монополізація ринку Інертний ринок

Продовження таблиці 5.12

Можливості:	Загрози:

Інвестиції Реалізація лабораторного стенду. Висока зацікавленість інститутів які вивчають мережі	Застарілі системи на підприємствах де не можна буде ввести данни стенд Вибір споживачів на користь перевірених рішень

Визначені альтернативи були проаналізовані з точки зору термінів та ймовірності отримання ресурсів. Результати аналізу наведені у таблиці 5.13.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Маркетингова кампанія для приваблювання покупців	Ймовірне	3 місяці
2	Безкоштовне впровадження системи інтеграції в одного із покупців	Дуже ймовірне	1 місяць
3	Пошук підприємств та інститутів які хотіли би впровадити інтеграційну систему	Ймовірне	4 місяців
Обрана альтернатива: Безкоштовне впровадження системи інтеграції в одного із покупців			

5.5 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів. Виконаний опис наведено у таблиці 5.14.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Банкові установи	Висока	90%	Є декілька конкурентів	Високі бар’єри входу
2	Рітейл організації	Висока	60%	Є декілька конкурентів	Середні бар’єри входу
3	Промислові підприємства	Середня	50%	Є декілька конкурентів	Низькі бар’єри входу
Які цільові групи обрано: рітейл організації					

За результатами аналізу потенційних груп споживачів (сегментів) була обрана цільова група рітейл організації для впровадження у яких буде пропонуватись система, та визначили стратегію охоплення ринку, яка наведена у таблиці 5.15.

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Встановлення інтеграційної системи у рітейл організаціях, а також у підприємствах які потребують дану систему	Концентрований маркетинг	Низькі витрати Ефективна співпраця посередників	Стратегія диференціації

Обраною базовою стратегією розвитку є стратегія диференціації, яка передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмінним від товарів конкурентів. Така відмінність базується на відсутності аналогічних рішень.

Виходячи з активного впровадження ІТ-сервісів у державі, було визначено базові стратегії конкурентної поведінки, наведені у таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Залучати нових	Ні	Стратегія заняття конкурентної ніші.

Важливим при виході на ринок є впізнаваність бренду, чому сприяє формування комплексу асоціацій, за якими користувачі можуть однозначно ідентифікувати продукт.

Для формування ринкової позиції (комплексу асоціацій), за яким споживачі будуть ідентифікувати проект, було визначено стратегію позиціонування, наведену у таблиці 5.17.

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)

Продовження таблиці 5.17

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Простота у користуванні Інтегрованість різних джерел інформації Захищеність інформації Велика кількість готових конекторів для роботи зі всіма популярними джерелами інформації Моніторинг	Стратегія диференціації	Формування регулярного попиту Збільшення разового використання послуги Виявлення нових груп споживачів Нові напрями застосування існуючої послуги	Оптимальна ціна, задоволення потребам, якість та надійність

5.6 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач.

У таблиці 5.18 підсумовані результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Створення системи інтеграції гетерогенних джерел даних	Гнучкість в роботі з даними, доступу до даних організації Низькі ціни	Цінова політика. Впевненість у професійності підготовлених спеціалістів
2	Навчання розгортання та адміністрування працівниками інтеграційної системи	Оперативність в розгортанні та адмініструванні системи	Конкуренти пропунують підтримку, але за цю можливість необхідно сплачувати невиправдано велику суму
3	Управління хмарною IT інфраструктурою	Простота зв’язку з автором роботи Можливість ознайомлення з іншими роботами автора	Можливість керувати великою кількістю сервісів, та інтегрувати нові

Далі за уточненими характеристиками продукту була розроблена трирівнева маркетингова модель товару, яка наведена у таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
І. Товар за задумом	Система інтеграції гетерогенних джерел даних
	Властивості/характеристики
	1. Надійність 2. Зручність 3. Швидка обробка данних
	Якість: коректність розгортання програми
	До продажу: Навчання користувачів
	Після продажу: реалізація додаткових сервісів
За рахунок чого потенційний товар буде захищено від копіювання: підписанням умови про нерозголошення	

Далі були оцінені принципи ціноутворення на ринку, за результатами чого були визначені межі встановлення ціни, наведені у таблиці 5.20.

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	—	30000 – 50000 грн	1000000 грн	10000 – 25000 грн

Розрахована оптимальна система збуту наведена у таблиці 5.21.

Таблиця 5.21 – Оптимальна система збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Закупівля здійснюється через відкриті тендери	Створення тендерної документації Інформування користувачів	Канал одного рівня	Селективна з використанням комбінованого каналу збуту

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфікою поведінки клієнтів. Результати розробки концепції маркетингових комунікацій наведені у таблиці 5.22.

Таблиця 5.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Ведення діяльності в державних структурах,	Прямі офіційні	Послідовність в реалізації обраної позиції	Формування у цільової аудиторії обізнаності про появу нового сервісу	Раціоналістична стратегія реклами

Продовження таблиці 5.22

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Баньківських структурах, рітейл організаціях а також на приватних підприємствах		Доступність та об’єктивність інформації про фірму і товар Унікальність послуги Відповідність вимогам тендеру	Інформування користувачів про властивості та переваги сервісу Інформування користувачів про нові способи використання відомого сервісу Пояснення цільовій аудиторії принципу дії послуги Виправити у користувачів неправильні представлення про продукт	

ВИСНОВКИ

В магістерській дисертації розроблено систему інтеграції на основі WSO2. Із архітектури WSO2 використано такі сервіси як: WSO2 ESB, WSO2 DSS, WSO2 API-Manager.

У дисертації наведено обґрунтування та актуальність досліджуваної теми, розглянуті основні методи та підходи для побудови інтеграції між різними системами інформації та використано підхід з оглядом на використання наукових інструментів.

Розроблені структурна та функціональна схема системи інтеграції. Продемонстрована взаємодія між сервісами всередженні архітектури, а також відображено на блок-схемах життєвий цикл процесу, послідовність використання медіаторів, взаємодія їх між собою.

Продемонстровано процес розгортання системи, приклад якої дозволить реалізовувати подібні інтеграційні рішення на різноманітних підприємствах, а саме на підприємства які мають гетерогенні джерела даних.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) [Електронний ресурс] – Режим доступу до ресурсу: https://wso2.com/enterprise-integrator/6.5.0.
2) [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/cloud/high-speed-data-transfer
3) [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/cloud/secure-gateway
4) IBM App Connect Enterpise Info [Електронний ресурс] – Режим доступу до ресурсу: https://developer.ibm.com/integration/docs/app-connect-enterprise/.
5) System architecture [Електронний ресурс] – Режим доступу до ресурсу: https://developer.ibm.com/integration/.
6) [Електронний ресурс] – Режим доступу до ресурсу: https://info.lightwellinc.com/blog/app-connect-enterprise-as-an-api-microgateway.
7) Mule ESB [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/MuleESB.
8) [Електронний ресурс] – Режим доступу до ресурсу: http://www.sovtex.ru/products/esb/.
9) OpenStack [Електронний ресурс] – Режим доступу до ресурсу: https://wso2.com/integration/micro-integrator/.
10) Приклад архітектур [Електронний ресурс] – Режим доступу до ресурсу: https://docs.wso2.com/display/ESB490/Architecture.
11) [Електронний ресурс] – Режим доступу до ресурсу: https://wso2.com/whitepapers/wso2-enterprise-integrator-7-0-the-revolutionary-cloud-native-hybrid-integration-platform/.
12) [Електронний ресурс] – Режим доступу до ресурсу: https://ei.docs.wso2.com/en/latest/micro-integrator/references/synapse-properties/data-services/.

13) [Електронний ресурс] – Режим доступу до ресурсу: https://wso2.com/whitepapers/a-guide-to-wso2-identity-server/#03.
14) [Електронний ресурс] – Режим доступу до ресурсу: https://ei.docs.wso2.com/en/latest/micro-integrator/develop/creating-projects/.